1-1-1994

REPRESENTATION OF COLOR AND SEGMENTATION OF COLOR IMAGES

Margaretha Schwarz
Purdue University School of Electrical Engineering

Lynne Grewe
Purdue University School of Electrical Engineering

Avi Kak
Purdue University School of Electrical Engineering

REPRESENTATION OF COLOR

AND

SEGMENTATION OF COLOR IMAGES

Margaretha Schwarz
Lynne Grewe
Avi Kak

TABLE OF CONTENTS

REPRESENTATION OF COLOR AND SEGMENTATION OF COLOR IMAGES

ABSTRACT

We have discussed the various schemes used today for representing color, brought out the relationships between them, and, finally, discussed the edge and region based segmentation of color images. We have compared some commonly used segmentation techniques using different representations of color on real data. The report should prove useful as a first reading to students of computer vision interesting in the processing of color images.

1. INTRODUCTION

This report is a tutorial presentation of some of the more useful concepts in the representation of color and in the segmentation of color images. Although there exists a considerable body of literature on the subject, much of it is presented in a manner that leaves the reader with more questions than answers. We have found that it is often difficult for a student of computer vision to sort through the various schemes for color representation in order to see the relationships between them. The problem is compounded by the fact that the relationships between the various approaches to color representation are not mathematically clean, in the sense that these relationships, while possessing an edge of intuitive plausibility, can often not be defended on the basis of mathematically precise arguments.

Although the last two decades have witnessed some very impressive advances in computer vision, much of the progress has been confined to the use of non-color imaging for sensory input. During this period, researchers restricted their attention to black and white images for two reasons: First, it seemed premature to tackle the more difficult problem of color vision while our thinking regarding what representations and control structures to use for scene interpretation was still in a formative stage. Second, good color cameras and the computer processing power needed for color vision have become commonplace only recently. For general references on the topic of computer vision the reader is referred to [1,181.

Consider the problem of computer vision for inspection of industrial parts. The following observation is critical to understanding why it is imperative to use color information for such inspection. Vision-based inspection calls for illuminating a scene with Some light source that is characterized by a distribution of energy over some range of wavelengths. Different wavelengths in this distribution are reflected with varying strengths from a given point in the scene, and then a vision sensor *integrates* the reflected light energy over some band of the wavelengths to produce the signal at a pixel (see Fig. 1). Due to the fact that integration

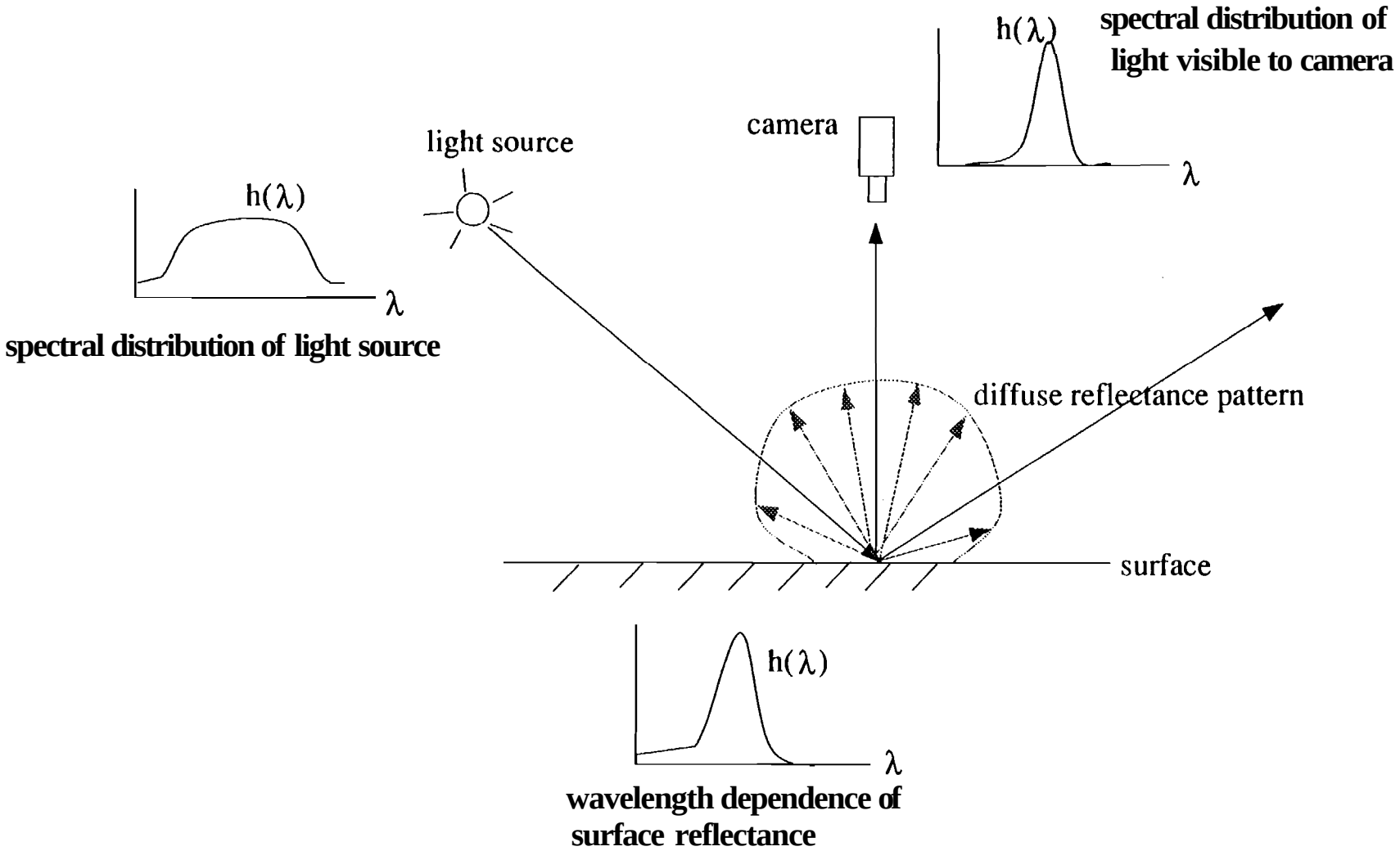


Figure 1 : The spectral distribution of light entering the camera as a function of scene illumination and surface reflectance.

over wavelengths is involved, surfaces with different reflectivity versus wavelength properties may produce the same signal strength at a pixel. For example, it is possible that two surfaces with different reflectivity versus wavelength properties, such as those shown in Fig. 2a and Fig. 2b, might produce exactly the same signal strength at a pixel for a vision sensor. Evidently, the more a sensor integrates over the spectra coming off a surface, the lesser the discriminatory power of the sensor. To the extent that a color sensor integrates over narrower bands of the spectrum, it would be more useful for distinguishing between surfaces.

To support this point, in Fig. 3 we have shown an image of a section of a surface-mounted circuit board as taken by a black and white camera. The component at the middle of the image, with code 000 printed on it, looks light green to the eye under broad-spectrum fluorescent illumination, while the surface of the board itself is dark green. As the reader can see from Fig. 3, it is practically impossible to distinguish between the component and the background. On the other hand, when the same board is imaged with a color camera, we get what is shown in Fig. 4 -- the component is now clearly distinguishable.

In the rest of this report, in Section 2 we will discuss some preliminary concepts relating to the experiential aspects of color and difficulties with providing engineering definitions of color. We then proceed to discuss the various possible spaces for the representation of color. Segmentation of color images will be presented in Section 3.

2. WHY IS COLOR REPRESENTATION SO DIFFICULT?

As has been established by psychophysicists over the past several decades, the information of color is fundamentally multidimensional. The three independent dimensions of color are hue, saturation, and intensity (HSI). Hue is what is usually referred to as color. From an engineering standpoint, hue is sometime thought of as the average wavelength associated with the light, but that is not entirely accurate. For example, if we optically combine the light emitted from a pure red laser with the light emitted from a pure blue laser, a human observer would perceive this combined light as magenta. On the other hand, if we could produce a spectrally pure light at a wavelength halfway between blue and red, that light would not be perceived as magenta. In other words, both the average wavelength and the shape of the spectral dismbution are important for characterizing hue.

The second independent dimension is saturation, also called chroma. If one had to give an engineering sense to saturation, one would say that a most saturated hue has no white in it; in other words, a hue will be called most saturated if its spectral components are only those that are ***necessary*** for producing the sensation of that hue. The saturation of a pure hue may be altered by the addition of white, in other words by the addition of a broad spectrum distribution to the dismbution that characterizes the pure hue.

The third independent dimension is intensity, also called brightness. Intensity refers to the light energy packed into the spectral distribution. For reflected light coming off a painting, the intensity may be reduced by adding a neutral medium like water to the color.

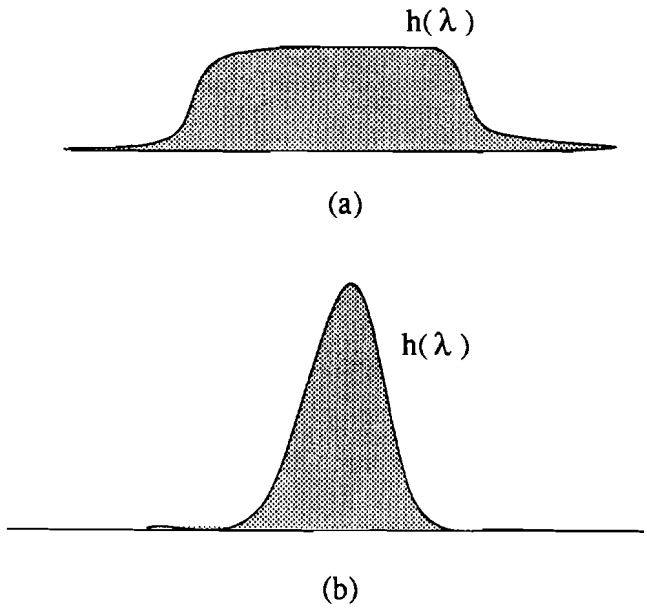


Figure 2: (a) The reflectivity distribution of surface 1. (b) The reflectivity distribution of surface 2. Notice that the areas under both curves are the same.

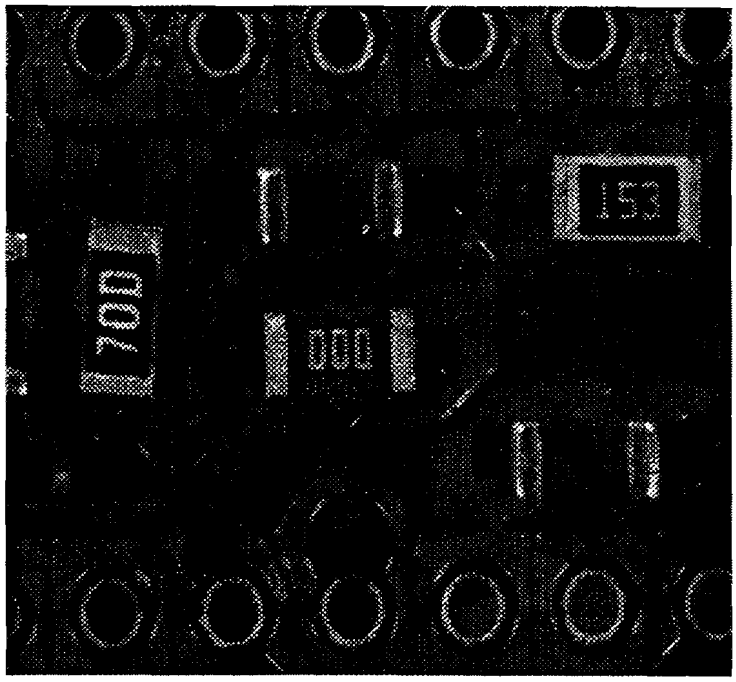


Figure **3:** A grey-scale image of a part of a circuit board.

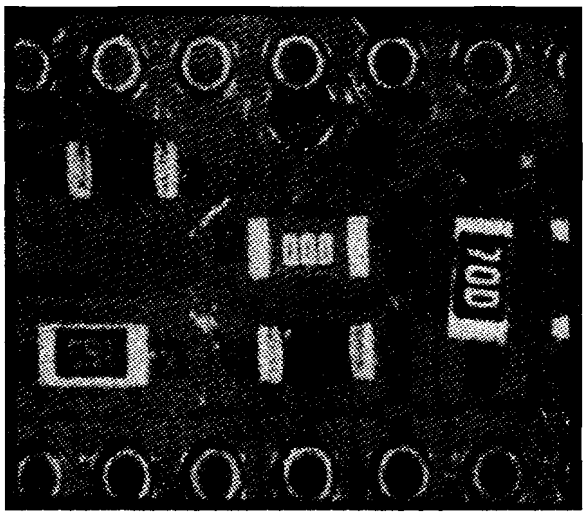


Figure 4: Color image of the scene shown in Fig. 3.

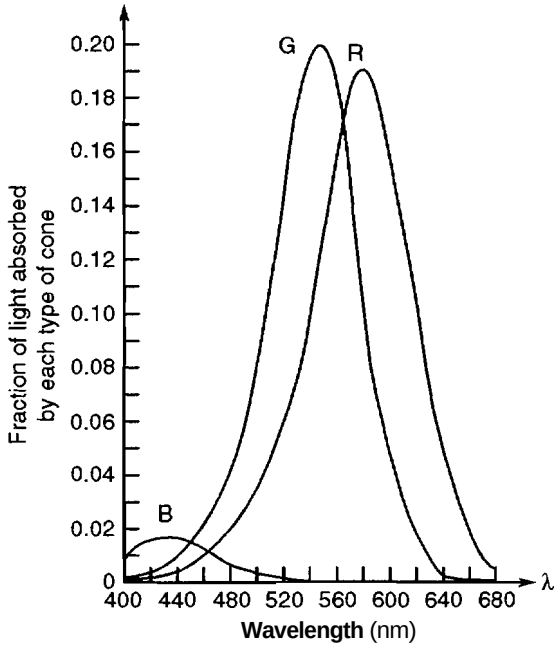


Figure 5: Spectral response functions of the three types of cones on the human retina. From Foley, **et.** al [3].

Although hue, saturation, and intensity describe more naturally the different aspects of color sensation experienced by humans, they are not as useful for designing engineering systems as some other descriptions. The more useful descriptions are based on the recognition of the fact that in the human retina there are three different receptors for color, each of these receptor types is characterized by a spectral dismbution of light to which it is sensitive (Fig. 5). If we refer to the three spectral distributions shown in Fig. 5 by $h_R(\lambda)$, $h_G(\lambda)$, and $h_B(A)$, and if the spectral distribution of illumination is given by $f_{ill}(A)$ and that of the object surface reflectivity by $f_{ref}(\lambda)$, we can write the following expressions for the signals sent by the three receptor types to the brain:

$$R = \int f_{ill}(\lambda)\, f_{ref}(\lambda)\, h_R(\lambda)\, d\lambda \qquad (1a)$$

$$G = \int f_{ill}(\lambda)\, f_{ref}(\lambda)\, h_G(\lambda)\, d\lambda \qquad (1b)$$

$$B = \int f_{ill}(\lambda)\, f_{ref}(\lambda)\, h_B(\lambda)\, d\lambda \qquad (1c)$$

The engineering implication of this is that if a vision sensor is to capture the true color -- in terms of the hue, saturation, and intensity experienced by a human -- then the vision sensor must record the three numbers R, G, and B as produced by these equations under the condition that the $h(\lambda)$ functions are as shown in Fig. 5. Since we may refer to the $h(\lambda)$ functions as filters, the vision sensor would have to filter the incoming light through three filters whose responses would be as shown in Fig. 5.

Since it is not practical to design optical filters with the kinds of responses shown in Fig. 5, approximations are necessary. The common approximation consists of using filters with "single line" responses for the three filters. That means the R filter passes through light at or in close vicinity to the 700 nm wavelength, the wavelength of the red hue. Similarly, the G filter passes through light at or in close vicinity to the 545 nm wavelength, the wavelength of green hue. And, the B filter passes through light at or in close vicinity to the 480 nm wavelength, the wavelength of blue hue. Mathematically, this commonly used engineering approximation is tantamount to saying that the filter functions $h_R(\lambda)$, $h_G(\lambda)$ and $h_B(\lambda)$ in Eq. 1 are replaced by delta functions $\delta(\lambda - \lambda_R)$, $\delta(\lambda - \lambda_G)$, and $\delta(\lambda - \lambda_B)$, respectively.

The primary justification for this approximation goes like this: Say we use a vision sensor based on the above approximation to represent a given color by three numbers R, G, and B. Now suppose we mix three spectrally pure light beams, one red, one green, and one blue, in the same ***proportion*** as the numbers R, G, and B. If a human observer sees this composite beam, the color experienced by the human would be ***roughly*** the same as the color in the original light beam on the vision sensor. The words most open to question in this justification are ***proportion*** and ***roughly;*** the former implies using only two degree of freedom in the synthesis of the composite beam and the latter is open to any interpretation. We will have more to say about these in the next section.

As it turns out, a number of colors cannot be synthesized by mixing red, green, and blue. This fact is made evident by Fig. 6 where it is shown how much red, how much green, and how much blue must be mixed in order to generate a spectrally pure color at a given wavelength. The significance to be associated with the wavelength band where the $\bar{r}_\lambda$

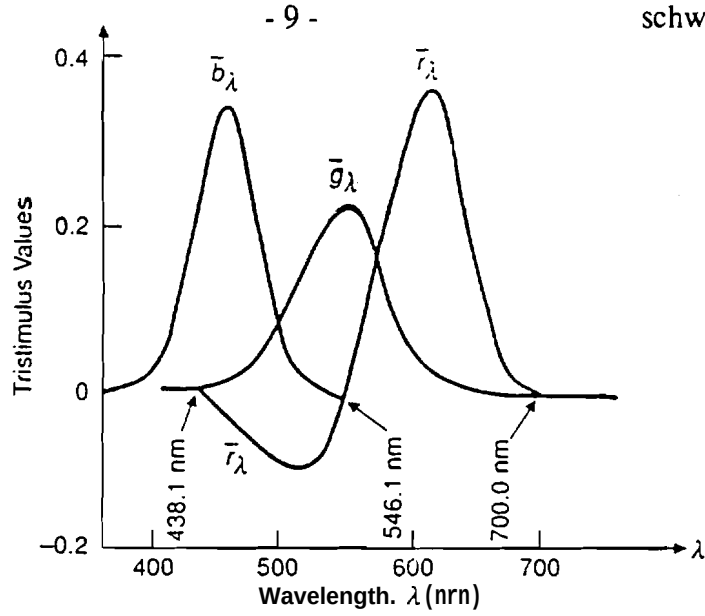


Figure *6:* Color-matching functions, showing how much red, how much green, and how much blue are needed to match all the wavelengths of the visible spectrum. The amount of a primary needed is with respect to a unit energy stimulus. From Foley, et. al[3].

becomes negative is most interesting. No spectrally pure color in this band can be produced by any combination of red, green, and blue. The negative values for $\bar{r}_\lambda$ mean that if we were to combine the red hue in proportion equal to the negative value of $\bar{r}_\lambda$ to the spectrally pure color, we could then match the combination with a mixture of blue and green.

Spectrally pure colors, although they represent only a fraction of all possible colors, are obviously of engineering significance since they can be generated with relative ease in laboratory for measurement and standardization. Since it is not possible to generate even all the spectrally pure colors by mixing red, green, and blue, these colors can really not be thought of as ***primaries.*** Therefore, three different colors, called simply **X,** Y, and Z, have been designated as primary colors; all the spectral colors in the visible band can be generated by a mixture of these primaries. Note for emphasis that ***only the spectrally pure colors can be generated by combining the*** X, Y, **Z** primary colors. It may not be possible to generate a non-spectral color, such as magenta, by combining the X, Y, **Z** primaries.

Our attempt to show the three primary colors **X ,** Y, and Z failed because the hardware of our display monitor is based on an **RGB** space. As will be explained in the next section, the three primaries **X ,** Y, and **Z** require a mixture of some negative values of R, G, and B which cannot be reproduced using a display hardware that is based on an RGB space.

3. COLOR SPACES

So far we have talked about characterizing color with three different systems, HSI, RGB, XYZ. Each of these systems amounts to representing color by a point in a three dimensional space. In this section, we will talk about the structure of these spaces and of some other related spaces. Which space one should use depends on the nature of the application. For electronic transmission and reproduction of color information and for color matching, spaces closely related to **RGB** and XYZ are used, while spaces related to HSI are preferred for applications that involve specification of color by humans, as for example in the design of synthetic imagery.

3.1 The Chromaticity Space

It is convenient to think of the space spanned by XYZ as constituting a three dimensional vector space. An arbitrary color, especially if it is a spectral color, may be represented by a point in this space. This is not to imply that the universe of all possible colors, as for example would be represented by all hues, all possible saturations, and all possible intensities, would be captured by this space, but since a large number of colors can be, the space is useful. Given an arbitrary color C, we may therefore represent it as

$$\mathbf{C} = X\,\mathbf{X} + Y\,\mathbf{Y} + Z\,\mathbf{Z}. \tag{2}$$

X, Y, Z being the components of the color along the three primaries. If the spectral distribution $P(\lambda)$ of a color C is known, the components X, Y, and Z may be found from

$$X = k \int P(\lambda)\, \bar{x}_\lambda\, \delta\lambda \tag{3a}$$

$$Y = k \int P(\lambda)\, \bar{y}_\lambda\, \delta\lambda \tag{3b}$$

$$Z = k \int P(\lambda)\, \bar{z}_\lambda\, \delta\lambda \tag{3c}$$

where $\bar{x}_\lambda$, $\bar{y}_\lambda$, and $\bar{z}_\lambda$ are the functions of wavelength as shown in Fig. 7, and k is a value that depends on the device. For example, a self-luminous object like a CRT has a k value = 680 lumens/watt. Using vector notion, we may therefore say

$$\mathbf{C} = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \tag{4}$$

We will now introduce the notion of chromaticity space. Representation of color in chromaticity space is based on an empirically verifiable premise that a large number of colors can be produced by mixing the X, Y, and Z in just the right proportions, without regard to their absolute values. To make this notion more precise, consider the following

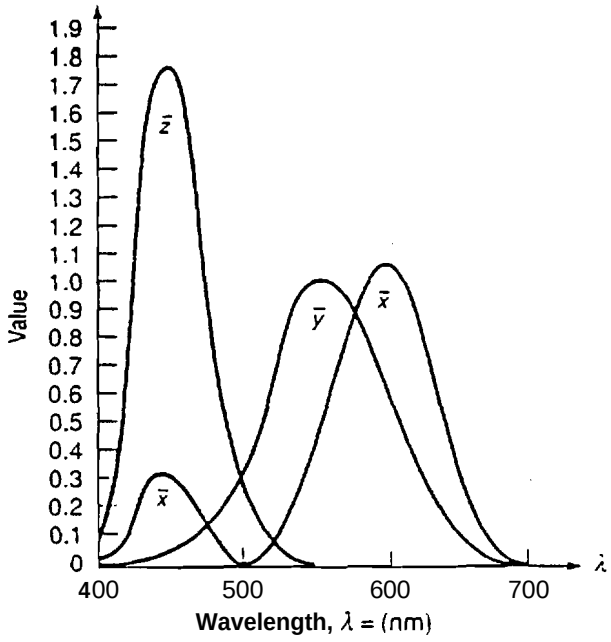


Figure **7:** The color-matching functions $\bar{x}_\lambda$, $\bar{y}_\lambda$, and $\bar{z}_\lambda$ for the X, Y, and Z primaries. From Foley et. al[3].

normalized versions of X, **Y** , and **Z:**

$$x = \frac{X}{X+Y+Z} \tag{5a}$$

$$y = \frac{Y}{X+Y+Z} \tag{5b}$$

$$z = \frac{Z}{X+Y+Z} \tag{5c}$$

With this normalization, **x,** y, and z will obey the following constraint

$$x + y + z = 1. \tag{6}$$

What is of engineering significance here is that as long as X, **Y** and **Z** are in the same ratio vis-a-vis one another, the values of x, y and z will remain unchanged and the constraint will be satisfied. To give the reader greater insight into the transformation from **XYZ** to xyz, consider the following two dimensional case:

$$u' = \frac{u}{u+v} \tag{7a}$$

$$v' = \frac{v}{u+v} \tag{7b}$$

Just by substitution, the reader can easily verify that for all the points along the line OA in the uv space, the corresponding u', v' value will be at A' (Fig. 8). Similarly, all the points on

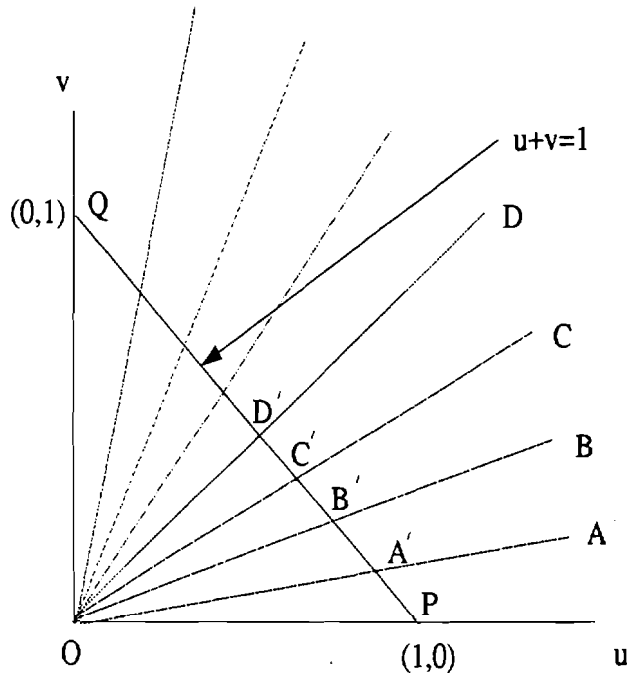


Figure 8: When $u' = \frac{U}{u+v}$ and $v' = \frac{V}{u+v}$, the points (u',v') will all lie on the line PQ for all points (u,v) in the positive quadrant of the uv-plane.

the line OB will be transformed into a single point B', and so on. In other words, the entire positive quadrant in the uv plane will be transformed into the line PQ. Mathematically, this fact is inferred readily from the observation that $u'+v'$ must equal 1.

Extending our two dimensional example to the three dimensional case of **XYZ** (Fig. 9a), we see that for all the values of X, Y, and Z the value of x, y, and z will be confined to the planar surface denoted by P, Q and S in the Fig. 9a. This plane will be referred to as the ***chromaticity plane.*** The projection of this planar surface on one of the orthogonal planes constitutes the ***chromaticity diagram.*** Usually, the projection on the XY plane is taken.

Also, it is more usual to think of the chromaticity plane directly in normalized coordinates, xyz, as opposed to the non-normalized coordinates XYZ. The only difference is that in the normalized coordinates, the space of points becomes confined to a unit cube in the positive quadrant of the XYZ space. In Fig. 9b, we have redrawn the chromaticity plane in normalized coordinates.

Remember the main justification for using x, y and z is that a large majority of colors can be produced simply by maintaining the right proportionalities between the three constituents X, Y, and Z. Translating into our diagram of Fig. 9b, that means that each point on the the planar surface bounded by P, Q, and S corresponds to some color. The colors

corresponding to the different points of this plane are shown in Fig. 10a. These colors may also be displayed by projecting the planar surface on P,Q,S onto the xy plane, as shown in Fig. 10b. In such displays of chromaticity diagrams, it is conventional to have boundary points correspond to those colors that would be produced by spectrally pure light. For example, if we were to take a tunable laser and change its wavelength continuously from one end of the visible spectrum to the other, as shown in Fig. 10b we would obtain colors corresponding to to a clockwise traversal of the boundary of the colors shown. So, the interior colors correspond to those that would require combining laser beams of different colors.

In Fig. 10b, the reader can see the points marked red, green, and blue. It is interesting to note that a mixture of any two colors that can be represented by points on the chromaticity plane will lie on a line joining those two points. Therefore, it follows that any mixture of red, green, and blue will result in a color that will correspond to a point in the mangle formed by the red, green, and blue points Fig. 10b. Such mangles are called ***color gamuts.*** It should therefore be clear from Fig. 10b, that a vision sensor or display device that represents colors by mixtures of red, green, and blue will certainly not cover all the colors even on the chromaticity plane, a shortcoming exacerbated by the fact that there are other colors that are not even represented on the chromaticity plane.

Before concluding the discussion on chromaticity space, we must mention that it is conventional to refer to colors by their coordinates in usually the xy plane. As it turns out, different devices produce the same color at slightly different locations on the chromaticity plane. For example, as pointed out by Foley et al. [3], short-persistence phosphors produce the red, green, and the blue colors at the normalized coordinates:

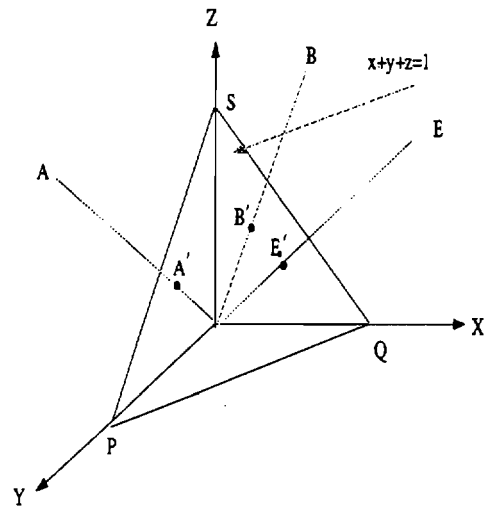
	Red	Green	Blue
x	0.61	0.29	0.15
y	0.35	0.59	0.063

while the long-persistence phosphors produce the same colors at:

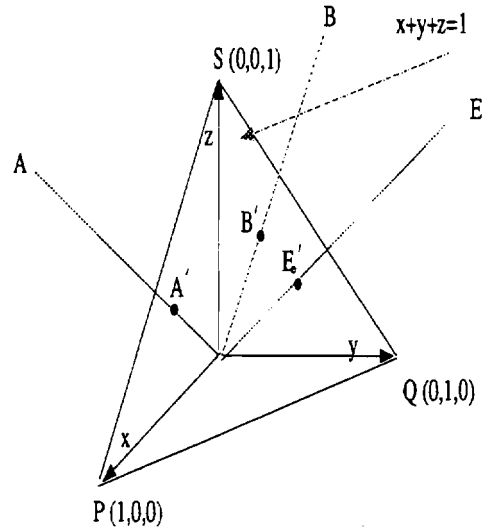
	Red	Green	Blue
x	0.62	0.21	0.15
y	0.33	0.685	0.063

3.2 Representation of RGB as a Euclidean Space

As mentioned in the Introduction, hardware for capturing and displaying color images uses the RGB representation, meaning that each color is expressed by a mixture of red, green, and blue; this is despite the fact that the colors represented by mixtures of RGB are only a subset of the colors that can be generated in the X, Y, Z space. Therefore, if an electronic camera is used as a vision sensor -- as we do in our research -- the pixel values will



(a)



(b)

Figure 9: (a) The points (x,y,z), when related to (X,Y,Z) by Eq. (5.5), will always lie on the triangular patch PQS. (b) Normalized representation of coordinates xyz for the chromaticity space.

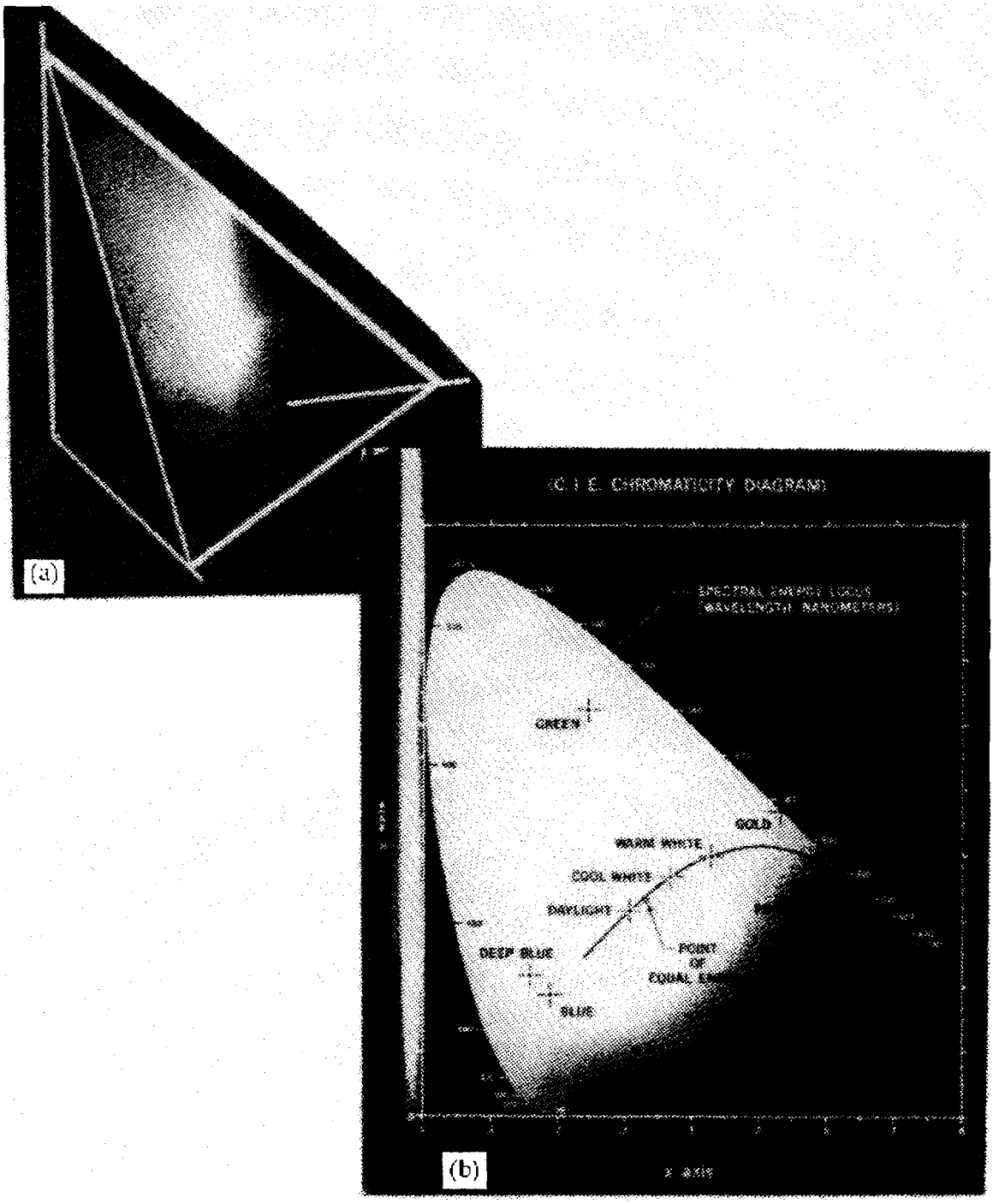


Figure 10: (a) Colors associated with the chromaticity plane. (b) The chromaticity diagram. The wavelength around the perimeter is in nanometers. From Foley. et. al **[3]**

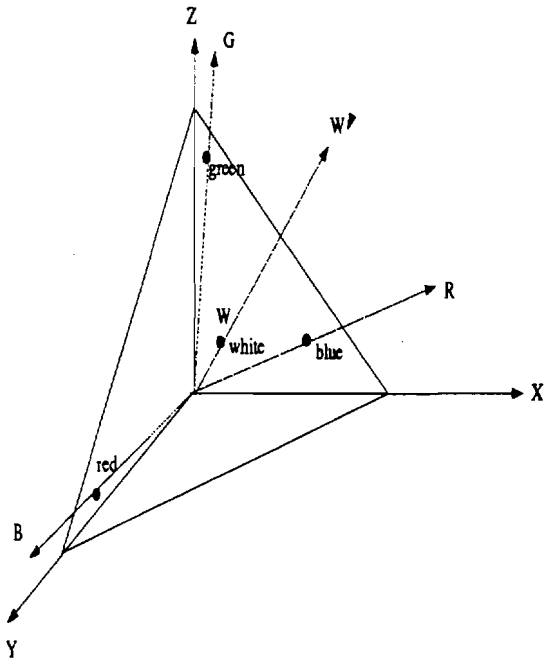


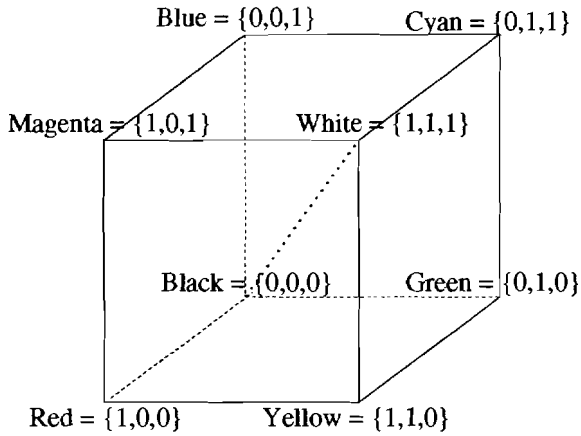
Figure 11: An illustration of the RGB space in the **XYZ** coordinate frame.

correspond to RGB coordinates.

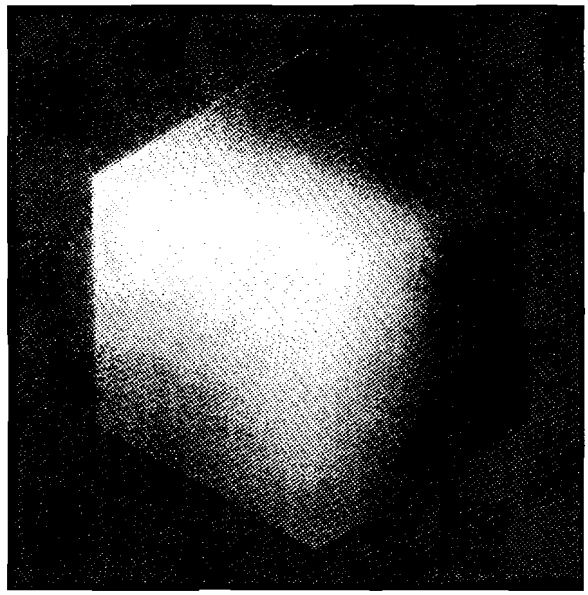
Strictly speaking, the RGB space is a subspace of the XYZ space. This fact is made clear by Fig. 11 where we have drawn the X, Y, and Z vectors in an orthogonal frame; the vectors corresponding to red, green, and blue are then shown by drawing lines through the corresponding points on the chromaticity plane. It may seem that the RGB vectors would span the same space as the XYZ vectors -- and mathematically that is true -- but because colors can only be made from positive mixtures, one can see the positive quadrant of the RGB space would be a subset of the positive quadrant of the XYZ space.

Despite the non-orthogonality of the RGB vectors, especially in relation to the XYZ space, for visualization purposes it is convenient to display the RGB space as consisting of orthogonal basis vectors R, ***G***, and B (Fig. 12). As shown in the figure, the space spanned by R, G, and B is displayed as a unit cube, the vertex (1,1,1) corresponding to the intensities of red, green, and blue that it takes to produce white.

It may seem like a contradiction that while the colors white, red, green, and blue are coplanar in Fig. 11, they are non-coplanar in the RGB cube of Fig. 12. This contradiction is not real for the following reason: In the chromaticity depiction, every color is forced to lie on the chromaticity plane of Fig. 11 whether or not it really does. In other words, the chromaticity coordinates only give us the relative proportions of the primaries in a mixture but not their true values. So, it may well be that to produce the color white we may have to



(a)



(b)

Figure 12: (a) An RGB space displayed as a unit cube (b) The colors associated with the space when viewed along the main diagonal. From Foley, et.al. [3].

be at the point W', but, as far the chromaticity depiction is concerned, the color white will be represented by the point W.

Fig. 12b shows the colors on the surface of the RGB cube as the cube is viewed inwards from the positive quadrant.

Although not directly useful in everyday color processing, one should note that linear transformations are available that readily convert from XYZ values into RGB values and vice versa.* The relationship between the XYZ space and the RGB space can be expressed as :

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} 0.618 & 0.177 & 0.205 \\ 0.299 & 0.587 & 0.114 \\ 0.000 & 0.056 & 0.944 \end{bmatrix} \cdot \begin{bmatrix} R \\ G \\ B \end{bmatrix} \tag{8}$$

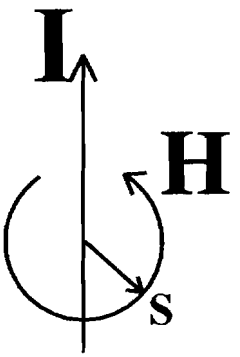
This is a transformation between the X, Y, Z, space and optically standard R, G, B colors. Since R, G, B for a practical device, such as a lab monitor, might be different from the optical standard, the correct transformation between the two spaces must be derived, as discussed in Chapter 13 of [3]. The color output of two different R, G, B devices can be compared after the transformation from each to the X,Y,Z space is known.

3.3 Hue/Saturation/Intensity Space

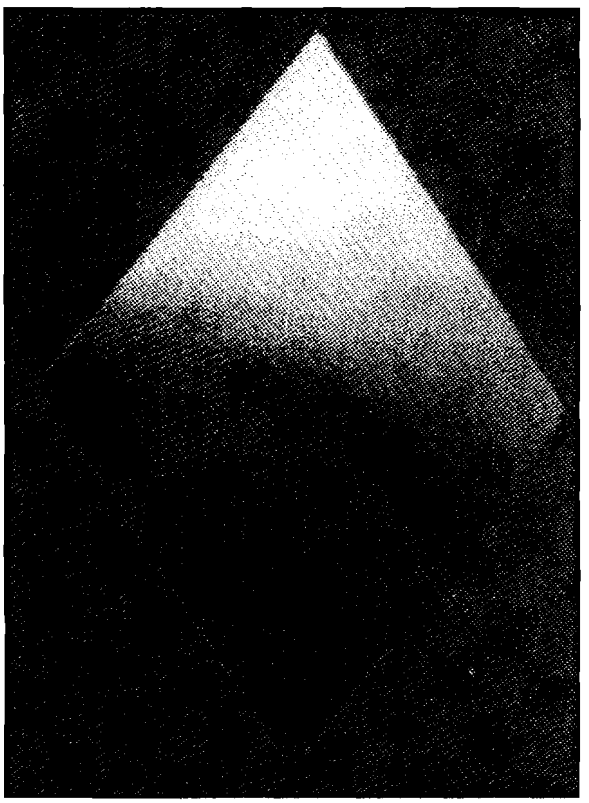
We have already discussed the HSI scheme for representing colors. The HSI coordinates are usually shown in cylindrical coordinates, with H being represented by the azimuthal angle measured around the vertical axis, S by the outward radial distance, and I the height along the vertical (Fig. 13). The red hue is placed at $H = 0^0$, the green at $H=120^0$, and the blue at $H=240^0$. The point I=1 on the vertical axis corresponds to the white color, and the origin to the black color since the intensity will be zero there.

Since HSI coordinates are most accessible to human intuitions about color and since RGB is what is used by most electronic devices for image capture and display, it is important to establish a transformation between the two spaces. We will now show two different approaches to this transformation.

* The reader might ask if a linear transformation can convert any XYZ value into a corresponding RGB value, and if it is possible to generate all spectral colors by mixing XYZ primaries, why it is not possible to generate all such colors with mixtures of R, G, and B. The answer is that there is no guarantee that such transformations would yield positive weights for the RGB colors.



(a)



(b)

Figure 13: The HSI space. (a) Interpretation of the components of HSI space (b) The colors associated with the HSI space. From Foley, et. al [3].

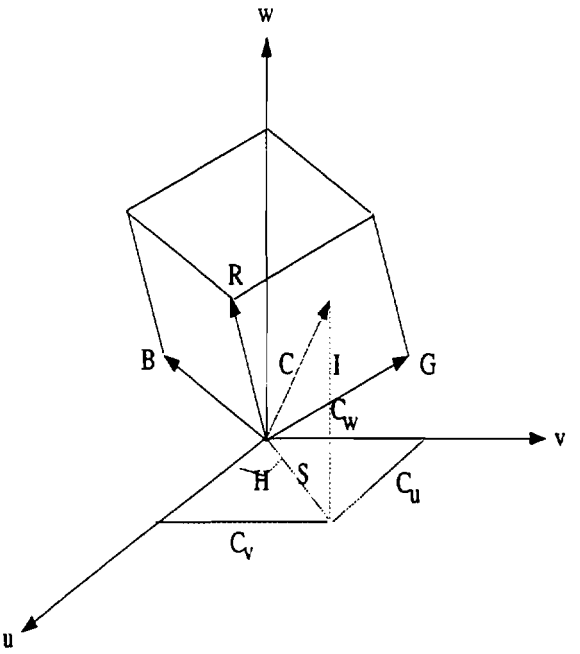


Figure 14: An illustration of the transformation from RGB to HSI space.

Transformation based on RGB cube rotation

To set up a transformation between the two, we line up the diagonal of the the RGB cube in Fig. 12a with the vertical axis of Fig. 13, since we want the the (1,1,1) point of the RGB cube to become coincident with the 1=1 point on the vertical axis of the HSI cylindrical representation, both these points representing the white color (Fig. 14). Furthermore, the rotational orientation of the cube is such that the red axis of the cube is in same azimuthal plane (the H=0 plane) that contains the red hue on the HSI cylinder. This will automatically cause the green axis of the RGB cube to fall on the azimuthal plane corresponding to $H=120^0$; recall the point at $H=120^0$, S=1, and 1=1 designates green in the HSI system. Similarly, for the blue axis of the RGB frame.

One may now write down the transformation equations that take a color point from RGB to HSI or vice versa. To facilitate the derivation of these equations, let's momentarily assume that the HSI system is embedded in a Cartesian frame whose axes are labeled U, V and W, as shown in Fig. 14. Consider now an arbitrary color point C, whose projections on the U, V, W axes are given by C_u, C_v, and C_w. Clearly, then the HSI values associated with this color point are given by

$$H = \tan^{-1} \frac{C_v}{C_u} \qquad (9a)$$

$$S = \sqrt{C_u^2 + C_v^2} \tag{9b}$$

$$I = C_w \tag{9c}$$

To express the transformation from RGB to HSI, all we now need do is express an arbitrary point C in the RGB points in the UVW frame. Let the coordinates of this point in the RGB frame be given by C_r, C_G, and C_B and the corresponding coordinates in the UVW frame by C_u, C_v, and C_w. Then,

$$C_u = \sqrt{\frac{1}{6}(2\,C_r - C_b - C_g)} \tag{10a}$$

$$C_v = \sqrt{\frac{1}{2}(C_g - C_b)} \tag{10b}$$

$$C_w = \sqrt{\frac{1}{3}(C_r + C_b + C_g)} \tag{10c}$$

Since the arctan in Eq. 9a is a multivalued function, the value of H needs to be specified more precisely. Ambiguities are resolved by using the following set of formulas for H:

$$H = \cos^{-1}\left(\frac{C_u}{\sqrt{C_u^2 + C_v^2}}\right) \quad \text{if } C_v > 0 \tag{11a}$$

$$H = 360^0 - \cos^{-1}\left(\frac{C_u}{\sqrt{C_u^2 + C_v^2}}\right) \quad \text{if } C_v < 0 \tag{11b}$$

Since there is no guarantee that the value of S computed by Eq. 9b will not exceed 1, in actual practice Eq. 9b must be replaced by

$$S = \frac{S}{S_{max}} \tag{12}$$

which results in the following equation for S :

$$S = 1 - \frac{3 \min (C_R, C_G, C_B)}{C_R + C_G + C_B} \tag{13}$$

By substituting C,, C_v, and C_w in Eq. 9 with the corresponding expressions from Eq. 10, we will get the relationship between HSI and RGB coordinates.

Although useful in an approximate sense, there are serious conceptual shortcomings in the above derivation of the transformation. First and foremost, we assumed the RGB space to be orthogonal. As was mentioned in Section 1.2, while the R, G, and B vectors may be linearly independent in the XYZ space, they are certainly not orthogonal. Their orthogonality in Fig. 12 was meant simply to make easier the visualization of color distributions.

So, strictly speaking, to derive a transformation between the RGB space and HSI space, one must derive transformation equations that relate the RGB vectors, as shown in Fig. 11, with the HSI space as shown in Fig. 13, the transformation being controlled on the one hand by the alignment of the white points in the two spaces and, on the other, by the requirements that the R, G, and B vectors in Fig. 11 pass through the the designated points for the three colors in the HSI cylinder in Fig. 13. Such a transformation would be very complex for obvious reasons. However, it is possible to make reasonable approximations, as shown in Foley et. al. [3].

3.4 CMY Space

Understanding color in RGB space is important because electronic cameras produce R, G, and B signals and because display monitors take R, G, and B signals to output color images. Understanding color in HSI space is important because it is in that space we as humans understand color the best. When it comes to making hardcopies of color images, one must understand color in what is referred to as Cyan, Magenta, Yellow space or the CMY space.

Red, green, and blue are additive primaries, meaning that we may attempt to express any *emitted* color as an additive mixture of these three colors (Fig. 16a). On the other hand, cyan, magenta, and yellow are *subtractive* primaries (Fig. 16b). To explain what that means, shown in Fig. 15 is the color circle for pigments. We have shown some of the pigment hues on this circle; pigments corresponding to any two adjacent hues may be mixed to yield a pigment corresponding to the hue in between. For example, if we mix together the cyan and magenta pigments, we obtain the blue pigment (Fig. 16b).

When we say a pigment hue on the color circle of Fig. 15 is subtractive, what we mean is that if we shine white light on a surface coated with that pigment, it will absorb from the white light the hue at the *opposite* point on the color circle. This fact is made clearer with the illustration in Fig. 17. As shown there, assume that a surface is coated with the cyan pigment. Now assume that a beam of white light rich in all hues is incident on the surface, as depicted in the figure. The pigment will absorb the red hue and the distribution of hues in the reflected light will carry all hues except red. Since the reflected light must again be thought of as emitted light, we must add the hues on the color circle on the right to determine the color of the reflected beam. The hues that are opposite will add up to white. Therefore, the reflected light will bear the color of cyan, in addition to containing some white light. In other words, the reflected light will be an unsaturated version of cyan.

Just as a video monitor has separate guns for red, green, and blue, a hard copy printer has separate "pens" for cyan, magenta, and yellow. Suppose a pixel produced by a color

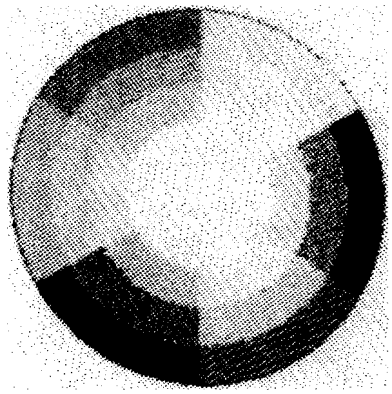


Figure 15: A color circle for pigments. From Foley, et. al[3].

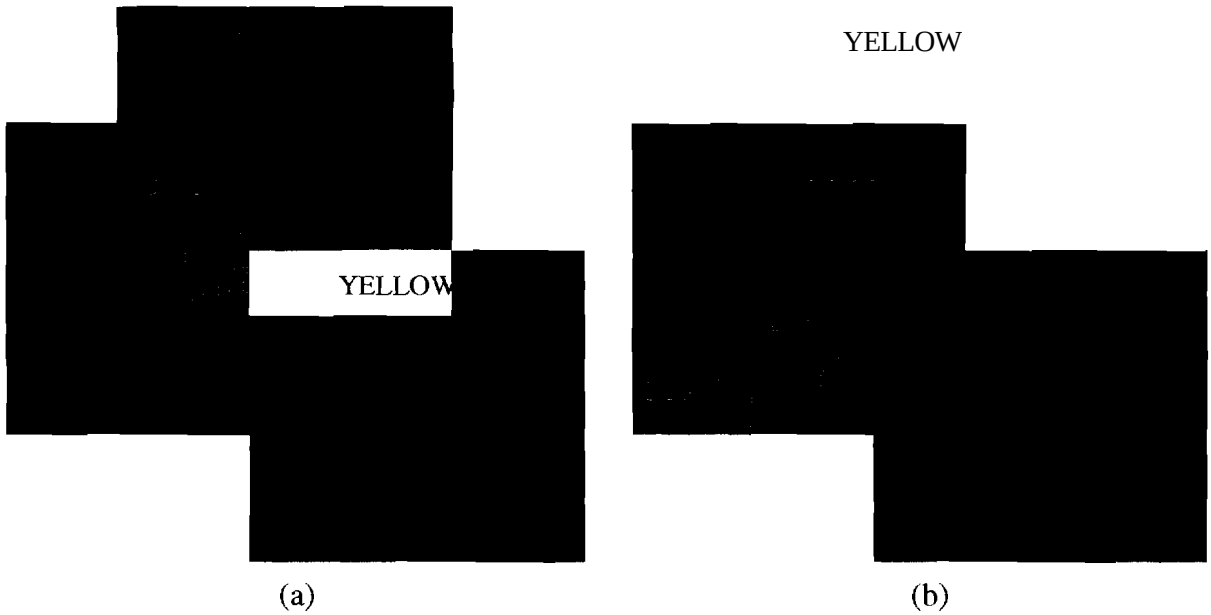


Figure 16: (a) Additive colors with Red. Green and Blue as primaries. Adding Red. Green, and Blue forms White, adding Green and Blue forms Cyan, etc. (b) Subtractive colors with Cyan, Magenta, and Yellow as primaries. Cyan and Magenta subtract Green and Blue from White, forming Red color, etc. From Foley, et. al [3].

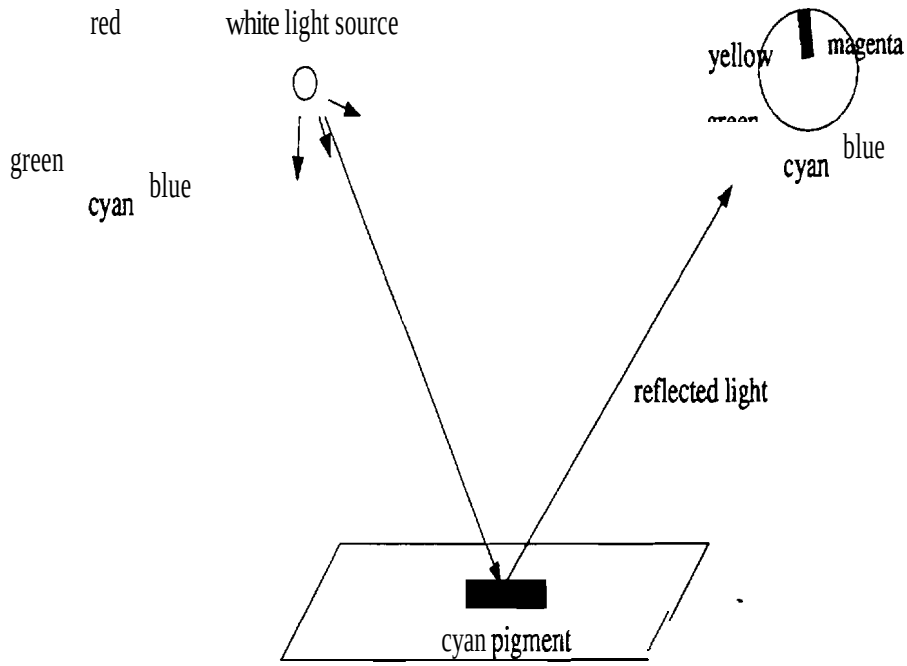


Figure 17: An illustration of subtractive color concept. A cyan pigment on the surface absorbs the red hue from the white light incident upon it and the reflected light lacks the red hue. From Foley, et. al [3].

camera has the following components of R, G, and B:

$$R = .8,\ G = .3,\ B = .4$$

When this pixel is displayed on a color calibrated video monitor, these RGB values will cause the monitor pixel to possess the right color. However, when the same pixel is sent for display to a hardcopy printer, the values sent to the cyan, magenta, and yellow pens must be

$$C = .2 = 1 - R$$
$$M = .7 = 1 - G$$
$$Y = .6 = 1 - B$$

C being 0.2 will cause the deposited cyan pigment to absorb 0.2 fraction of the red hue in the supposedly white illumination of the hardcopy surface, releasing 0.8 of the red hue in the direction of the observer. Similarly, with M and Y for the generation of the correct values of G and B for an observer.

This transformation between RGB and the CMY values required to produce the correct RGB in light reflected from a hardcopy is expressed succinctly by

$$\begin{bmatrix} C \\ M \\ Y \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} - \begin{bmatrix} R \\ G \\ B \end{bmatrix} \tag{14}$$

3.5 YIQ Space

Electronic color cameras frequently produce two different types of color signals, one is the RGB signal, which in effect consists of three separate signals, one for each primary, the other corresponds to a YIQ characterization of color. The YIQ characterization, like RGB, is also three dimensional but is modulated into a single continuous waveform, usually for broadcast purposes. In the US, this continuous waveform is referred to as the NTSC Composite signal.

The main rationale behind the YIQ characterization of color is that at least one component of color characterization should correspond to the brightness intensity variations in a scene if that scene were to be perceived with a monochrome camera, the resulting advantage being that by using just this component a color image may be displayed on a monochrome display device. The transformation between RGB and YIQ is given by

$$\begin{bmatrix} I \\ Q \\ Y \end{bmatrix} = \begin{bmatrix} 0.596 & -0.275 & -0.321 \\ 0.212 & -0.528 & 0.311 \\ 0.299 & 0.587 & 0.114 \end{bmatrix} \cdot \begin{bmatrix} R \\ G \\ B \end{bmatrix} \tag{15}$$

The reader will note that the mamx entries in the last row are the same as those corresponding to Y in the matrix shown in Eq. (8). This coincidence is by design, as the Y component of the YIQ characterization is meant to correspond exactly to the Y component of the XYZ space. The reason for this can be explained with the help of curves shown in Fig. 7; the $\bar{y}_\lambda$ function there matches exactly the brightness sensitivity of the human at different wavelengths. Therefore, if we compute the Y component of a light beam of a given spectral distribution by using Eq. 3b , the value we get is a measure of the brightness of the beam as experienced by a human observer.

Therefore, the Y component obtained via the above transformation gives us the brightness value, which, if necessary, can then be used by a monochrome camera to make a black and white image. Color information is encoded in the other two components, I and Q.

3.6 Linear Transformation Space

All of the spaces mentioned above have different uses depending on the applications. There are some other multidimensional spaces that are based on a transformation from the RGB space and that have been developed over the years for image segmentation. One effective transformation for image segmentation worth mentioning was developed by Ohta et al. [17]. The transformation was derived by analyzing the eigenvectors of the covariance mamx of the distribution of R,G, and B in eight different scenes. It is defined as :

$$I1 = \frac{R+G+B}{3} \tag{16a}$$

$$I2 = \frac{R-B}{2} \quad \text{or} \quad \frac{B-R}{2} \tag{16b}$$

$$I3 = \frac{2 \cdot G - R - B}{4} \tag{16c}$$

This transformation results in a set of independent color features I1, I2, and 13. They proved the effectiveness of these color features experimentally by comparing the segmentations obtained vis a vis the segmentations produced directly from the R,G,B space.

4. RECENT RESEARCH IN COLOR COMPUTER VISION

Surveyed below are some recent research papers that deal with the different aspects of color in computer vision:

Kender in [10] has analyzed the transformation of RGB into HSI, the normalized coordinates, the linear transformation space, and the YIQ space for natural scenes. He elaborated on the problems that arise in nonlinear transformations, especially the fact that such transformations are singular at some points and unstable at others, causing the transformed values to sometimes respond poorly to small variations in the RGB input. In addition, the digitization of RGB causes distortions in the distributions of the transformed output. Kender has shown that if the transformations are not used with care, the segmentations produced might be of inferior quality compared to the segmentations using RGB.

Nevatia [13] has developed a color edge detector by generalizing the Hueckel operator, derived for the monochrome case by the minimization of an error criterion, into a three dimensional space. He showed that the minimization step for all three components of a color image amounts to carrying out the minimization for each component independently. He concluded that the use of color would aid in detecting low luminance contrast edges and in extracting edges with higher confidence.

Ohta, Kanade, and Sakai [17] have experimented with color image segmentation using what is known as the {(L*,a*,b*)} features and the features shown in Eqs. (16). The L*,a*,b* space came into existence in 1976 [9]. The importance of the L*a*b* space lies in the fact that it is a uniform color space, meaning that the Euclidean distance separating two colors is proportional to their visual differences. The L*, a*, and b* dimensions are very akin to the orthogonal dimensions of the HSI space; these were denoted (U,V,W) in Fig. 14. The axis L* corresponds to the intensity I of the HSI system; the azimuthal planes pivoting around the L* axis correspond to constant H, and the cylindrical shells of constant radii

correspond to constant S. The transformation from from the RGB space to the L*a*b* space is accomplished by first using Eq. (8) to go from RGB to XYZ and then applying the following cube-root transformation:

$$L^* = 116\left[\frac{Y}{Y_0}\right]^{1/3} - 16 \qquad \left[\frac{Y}{Y_0}\right] > 0.01$$

$$a^* = 500\left[\left[\frac{X}{X_0}\right]^{1/3} - \left[\frac{Y}{Y_0}\right]^{1/3}\right] \qquad \left[\frac{X}{X_0}\right] > 0.01$$

$$b^* = 200\left[\left[\frac{Y}{Y_0}\right]^{1/3} - \left[\frac{Z}{Z_0}\right]^{1/3}\right] \qquad \left[\frac{Z}{Z_0}\right] > 0.01$$

where X_0, Y_0, and Z_0 are the XYZ values of the reference white. We may, for example, select for these 255 for 8-bit data representation of each color plane.

Celenk [2] has used a clustering approach for segmenting color images represented in the (L*,a*,b*) space. Celenk first constructs a one-dimensional histogram of just the hue component of the color values at each of the pixels. The peak regions of this histogram then allow him to place azimuthal boundaries on the different possible clusters in the L*a*b* space. Next, for all the pixels whose hues fall in one of the hue-bounded regions in the color space, Celenk constructs two one-dimensional histograms, one for just the saturation and the other for just the intensity. The peak regions in the saturation histogram give radial boundaries of the different possible clusters of pixels in the color space. Similarly, the peak regions in the intensity histogram (the L* histogram) delimit the clusters with respect to the third dimension. In this manner, Celenk is able to divide all the pixels in an image into a small number of categories on the basis of the hue, saturation, and intensity values. Resorting to one-dimensional histograms, as opposed to searching for clusters in the full L*a*b* space, results in considerable savings in the computational burden.

Novak, Shafer, and Willson [15] have analyzed the effects of camera nonlinearities on color images and have suggested corrections for the nonlinearities.

Color constancy is defined in [12] as the perceptual ability that permits us to discount spectral variation in the ambient light and assign stable colors to objects. In computer vision, the goal is to have the vision system perform this task. Recent work has been accomplished in the design of algorithms to achieve color constancy which in particular correct for specularity effects, the presence of shadows, and the color of the illumination source [4,5,7,8,11,12,16]. Some of these algorithms assume certain scene constraints, require more sensor measurements than the tuple [r,g,b] produced by common color camera to properly work, and may be computationally expensive.

Recently, Swain and Ballard [20] introduced a technique called the histogram intersection method for matching model and image histograms of multicolored objects. The histograms themselves are three dimensional, consisting of bins in the RGB space. Given a pair of histograms, histogram I for the image and histogram M for the model, each containing n bins, the intersection of the two histogram is defined to be

$$\sum_{j=1}^{n} \min \left[I_j, M_j \right]$$

It can be argued that the result of the intersection in the manner defined here is the number of pixels from the model that have corresponding pixels in the image. To obtain a fractional value between 0 and 1, the intersection is normalized by the number of pixels in the model histogram:

$$H(I,M) = \frac{\sum_{j=1}^{n} \min \left[I_j, M_j \right]}{\sum_{j=1}^{n} M_j}$$

As shown by the authors, if we scale the two histograms such that the total count in all the bins in both the model and the image histograms is the same:

$$\sum_{j=1}^{n} M_j = \sum_{j=1}^{n} I_j = N$$

it can then be shown that

$$1-H(I,M) = \frac{1}{2T} \sum_{j=1}^{N} | I_j - M_j | .$$

The function 1–H defines a distance metric, a scaled ***city-block*** **metric** for comparing a model histogram with an image histogram.

According to the authors, this method for recognizing objects is robust to scale changes, such as those introduced by an object being at different distances from the camera, but not scale invariant. Apparently, this method works particularly well if the color image of a scene is first processed by a color constancy algorithm to compensate for the effects of varying illumination. The authors have also developed a fast incremental version of this algorithm that allows real-time indexing into a large database of stored models using standard vision hardware. Another interesting development made by the authors is the histogram backprojection method that can be used to locate a model object in the image of a scene.

5. SEGMENTATION BASED ON COLOR

We will now discuss the application of edge operators to color images. We will also discuss region-based segmentation of color images by the split-and-merge method. The results obtained will be shown for the color image of Figure 4. Comparative results will be shown for the black and white image of Fig. 3.

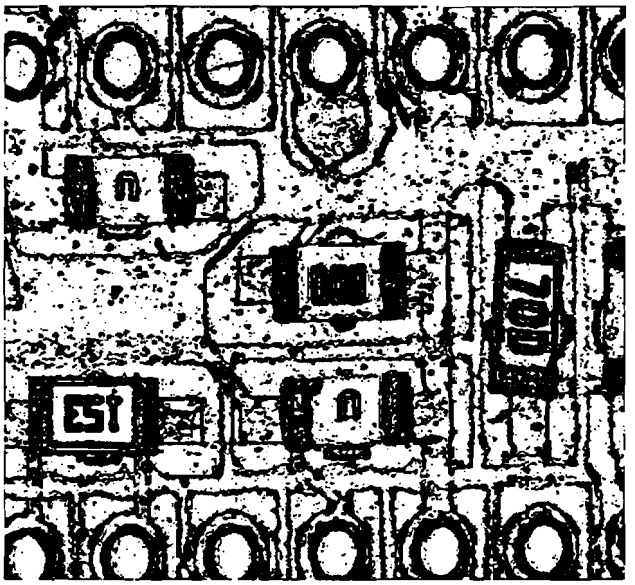
5.1 Color Edge Detection

We applied the usual Sobel edge detection operator separately to the R, G, and B values associated with each of the pixels of the image in Fig. 4. When so desired, the Sobel output for each color dimension was thresholded separately. The three Sobel outputs were combined disjunctively to produce the final edge image. Therefore, the final edge image shows an edge pixel if there is an edge pixel in any one of the three color dimensions. The result shown in Fig. 18a is the final edge image obtained in this manner after a threshold of 5 is applied separately to the R, G, and B edge outputs, meaning that an edge pixel is discarded if the Sobel magnitude there is less than 5 on a scale of 0 to 255. The result obtained after thinning the edges is shown in Fig. 18b. When in the edge outputs for each of the R, G, and B planes; the edges pixels where Sobel magnitude is less than 10 on a scale of 0 to 255 are deleted, the final result obtained by combining disjunctively all the edge ;planes is shown in Fig. 18c. After a thinning algorithms is applied to these edges, the result is shown in Fig. 18d.

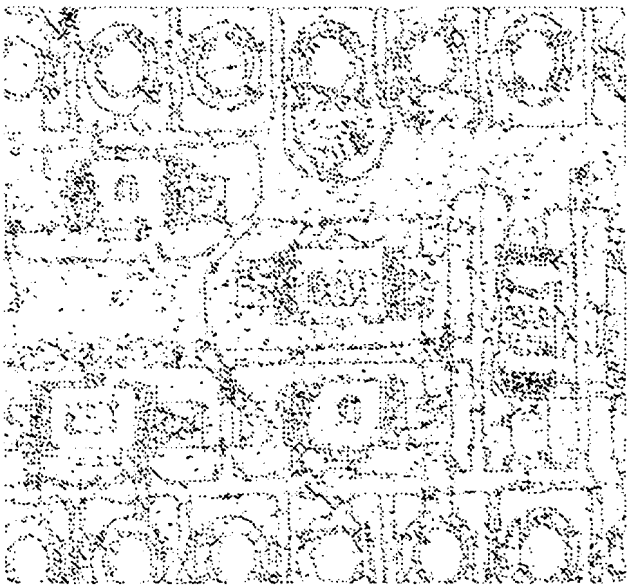
Less noisier edge images are obtained if the Sobel edge operator is applied not directly to the R, G, and B dimensions, but to the three dimensions that are obtained via a linear transformation of the RGB space. We are refemng to the linear transformation shown in Eq. (16). When the Sobel operator is applied to the I1, 12, and I3 dimensions separately, and edges pixels where Sobel magnitude is less than 5 on a scale of 0 to 255 deleted, we obtain the result shown in Fig. 19a after the edges images for I1, I2 and I3 are combined disjunctively. A thinned version of this output is in Fig. 19b. If the edge output for each of the I1, 12, and I3 dimensions is first thresholded in such a manner that pixels where the Sobel magnitude is less than 10 on a scale of 0 to 255 are deleted, the final edge output is as shown in Fig. 19c. A thinned version of this edge image is shown in Fig. 19d.

For comparison, we have shown in Fig. 20a a Sobel output for the black and white image of Fig. 3. Deleted from Fig. 20a are edge pixels where Sobel magnitude is less than 5 on a scale of 0 to 255. Shown in Fig. 20b are the thinned versions of the edges in Fig. 20a. Fig. 20c shows a similar result when the threshold is increased to 10, with the thinned version in Fig. 20d.

Comparing the results, we see that the edge results for the color images are noisier. This is not surprising because of the disjunction operation in the compilation of the final edge output. The use of the disjunction implies that a discontinuity in any of the color dimensions will cause an edge to appear in the final output. The edge output for the RGB space is worse compared to that for the I1,I2,I3 space. We believe that the latter representation results in superior edge output because of the energy compaction caused by the transformation used. Recall that the transformation from the RGB space to the I1,I2,I3 space is akin to a Karhunen-Loeve transformation.

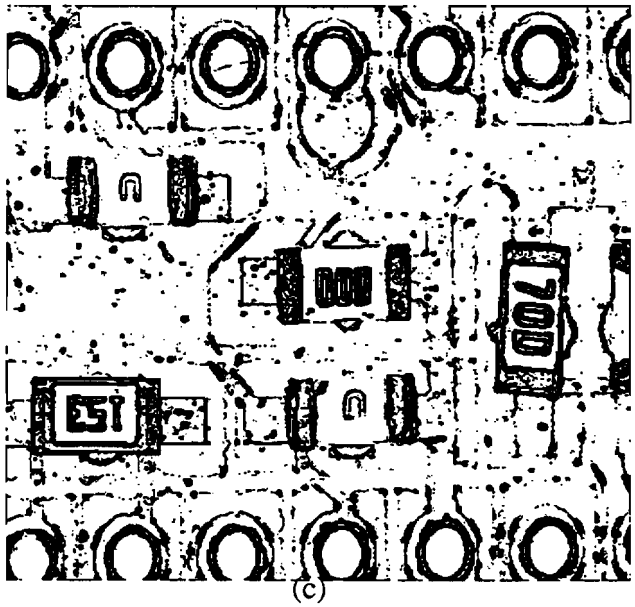


(a)

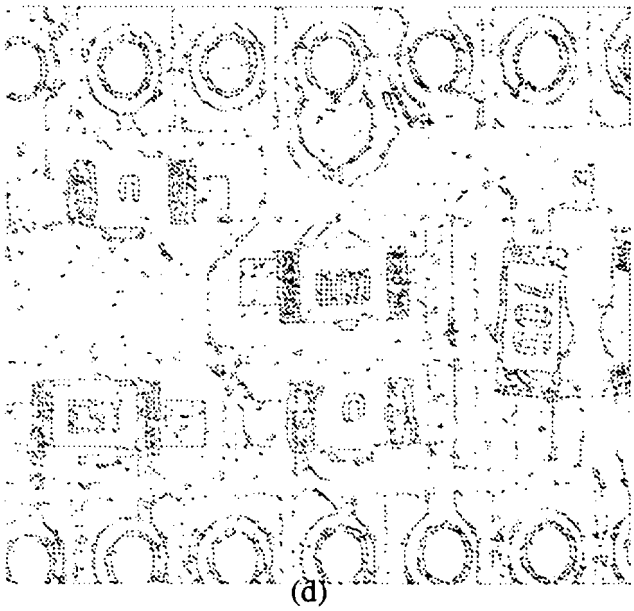


(b)

Figure 18,continued

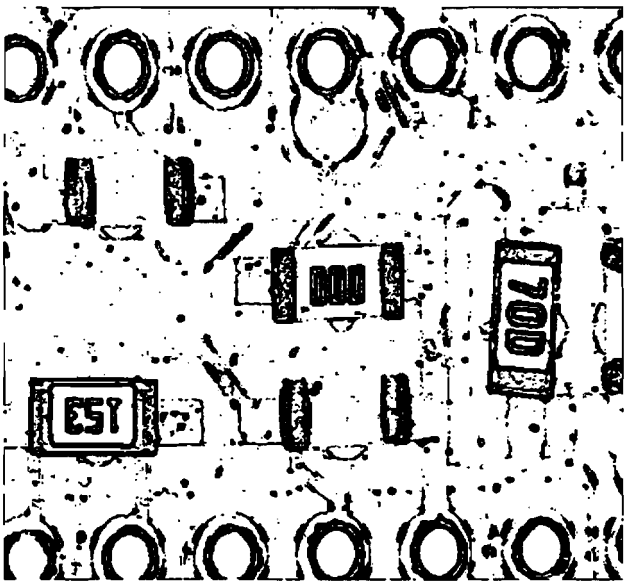


(c)

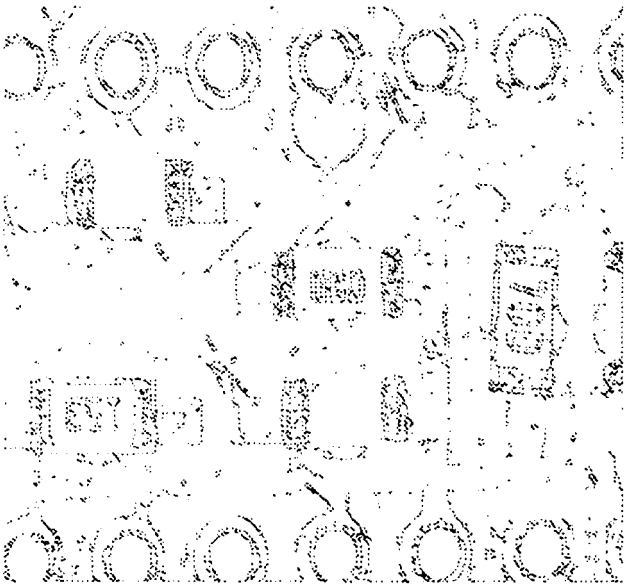


(d)

Figure 18: (a) This the final edge image obtained when a Sobel edge operator is applied separately to R, G, and B color planes for the image of Fig. 3. Edge pixels where the Sobel magnitude is less than 5, on a scale of 0 to 255, in all the color planes were deleted. (b) This is the result of thinning the edge image of (a). (c) Same as (a) except that a threshold of 10 was used in the three color planes. (d) The output obtained by thinning the edge image of (c).



(a)



(b)

Figure 19, continued

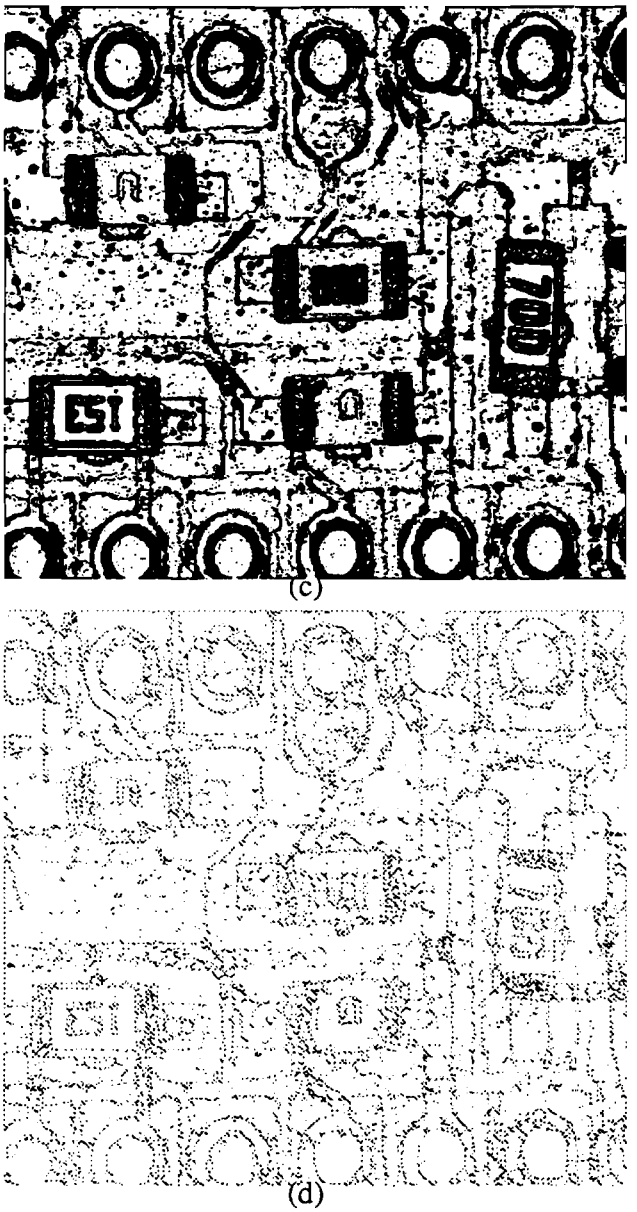
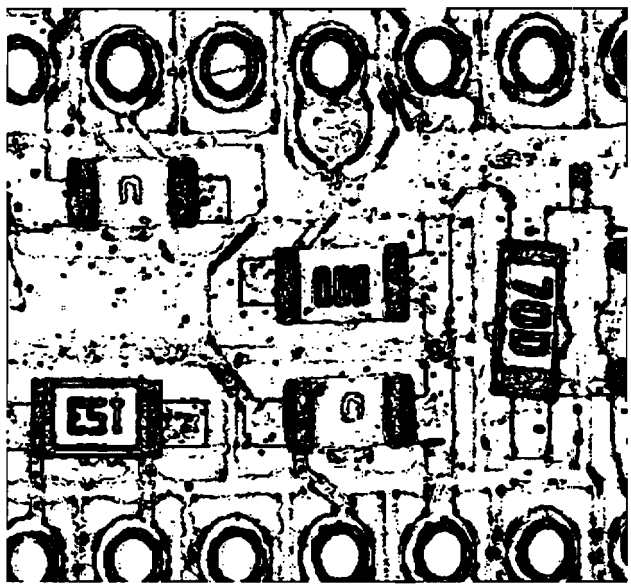
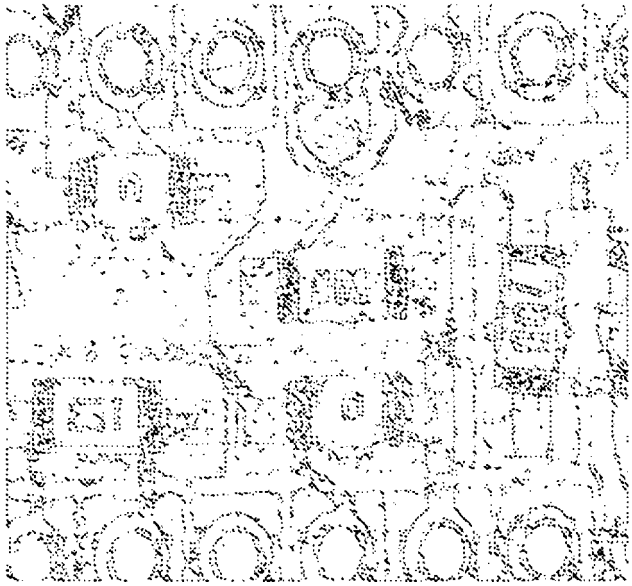


Figure 19: (a) This the final edge image obtained when a Sobel edge operator is applied separately to the I1, I2, and I3 color planes defined in Eq. (16). Edge pixels where the Sobel magnitude is less than 5, on a scale of 0 to 255, in all the color planes were deleted. (b) This is the result of thinning the edge image of (a). (c) Same as (a) except that a threshold of 10 was used in the three color planes. (d) The output obtained by thinning the edge image of (c).



(a)



(b)

Figure: 20, continued

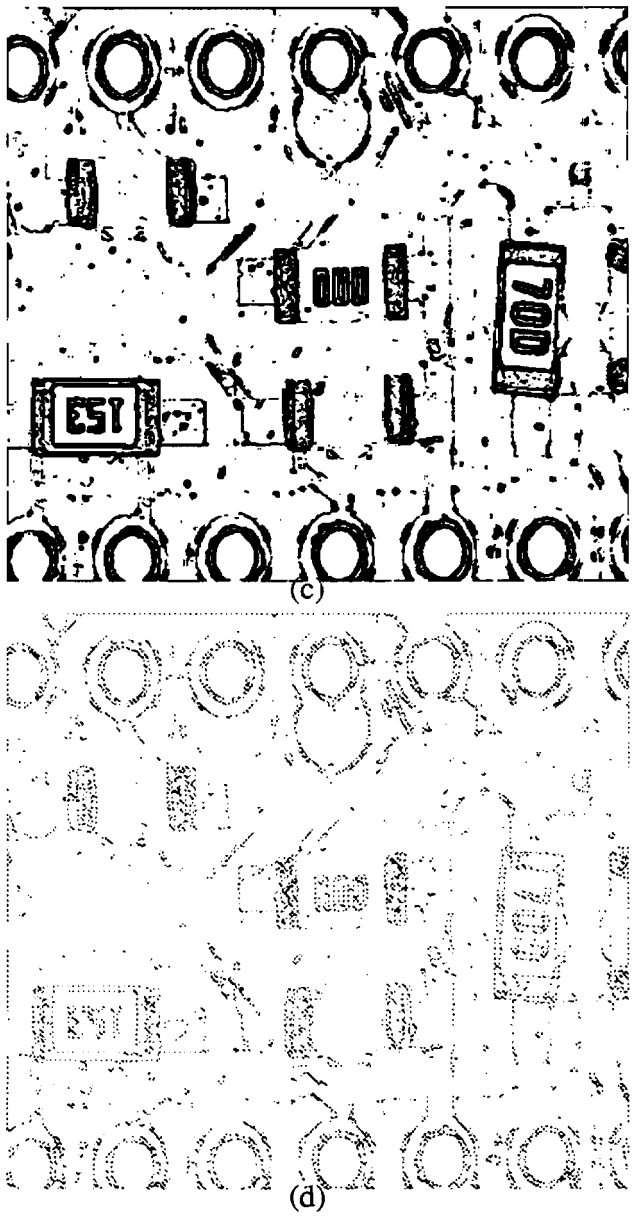


Figure 20: (a) The output of a Sobel operation on Fig. 3 with a threshold of 5. (b) Result of a thinning operation on (a). (c) The output of a Sobel operation on Fig. 3 with a threshold of 10. (d) Result of a thinning operation on (c).

5.2 Region-Based Segmentation Of Color Images By Split-And-Merge

The color split-and-merge method is an extension of the grey-scale split-and-merge. As the reader will recall, the gray-scale split-and-merge consists of the following steps:

STEP 1:

Build quadtree:

Build a quadtree representation of the image by recursively decomposing the image matrix into four quadrants, until each pixel is visited in a manner described by the Morton mamx [19]. Unwind the recursion, merging the pixels into nodes, and nodes into higher-level nodes, as long as all the pixels at each leafnode of the resulting quadtree satisfy the max-min constraint:

$$|\max f(x,y) - \min f(x,y)| \leq 2\varepsilon$$

In the resulting quadtree, each non-leafnode contains four pointers for its children.

STEP 2:

Sideways merging using max-min criterion:

The neighbor of a leafnode X is defined as an adjoining leafnode Y provided Y is not smaller than X in terms of the pixels contained (by adjoining is meant four-connected). With this definition, each leafnode will have a maximum of four neighbors. Use Samet's algorithm [19] to find all four neighbors, provided they exist, of every leafnode on the quadtree. The four pointers stored at a leafnode are used to point to its neighbors.

Recursively descend down the quadtree built in Step 1, until reaching the leafnodes.

If the region pointer of a leafnode is not set, instantiate a groupnode for the leafnode and set the region pointer of the leafnode to that groupnode. Now examine the neighbors of the leafnode currently being visited. If the neighbor can be merged with the leafnode on the basis of the max-min criterion, set the region pointer of the neighbor to the groupnode pointed to by the leafnode.

If the region pointer of the leafnode is set, recursively ascend the groupnodes pointed to by the region pointer, until reaching the root groupnode. Reset the region pointer of the leafnode directly to the root groupnode. Now examine the neighbors of the leafnode currently being visited. If the neighbor can be merged with the leafnode on the basis of the 2ε criterion of Step 1, and if the region pointer of the neighbor is not set, set the region pointer of the neighbor to the groupnode pointed to by the region pointer of the leafnode. On the other hand, if the region pointer of the neighbor is set, recursively ascend the groupnodes corresponding to that region pointer, until reaching the root groupnode. Make the root groupnode for the neighbor point to the root groupnode for the current leafnode if the tree hanging from the former is smaller than the tree hanging from the latter. Otherwise, do the opposite.

STEP 3:

Sideways merging using the average value criterion

The algorithm for merging the leafnodes here is exactly the same as in Step 2, except that now the criterion used for merging is different. Two neighboring leafnodes are merged, in the manner described above by the resetting of region pointers, provided the difference of the average gray levels of the pixels represented by the nodes is within some tolerance.

STEP 4:

Label the regions Descent recursively on the quadtree until reaching the leafnodes. For every leafnode thus reached, access the corresponding group node via the region pointer. Now ascend recursively the region pointers of the group nodes until reaching root. If the root does not already contain a region label, put an integer label there. For obvious reasons, this label will not be modified when the same root is accessed through some other leafnode.

STEP 5:

Merging of excessively small regions with adjoining large regions

The procedure for this merge is mostly the same as in Step 2, the major difference being that for the current quadtree leafnode we must first extract all its four neighbors before deciding whether there should be any merging at all and, in the event: one of the neighbors has gray-level properties that are close to the gray-level properties of the current leafnode, using the region pointers to recursively access the roots of the group nodes pointed to by the current leafnode and mergeable neighbor. If one of the regions corresponding to the two roots is too small, it is simply merged with the other region, assuming the size of the other region exceeds a threshold.

For extending this algorithm to color images, the max-min threshold (the 2s threshold) of Steps 1 and 2 and the threshold needed for average-value merging in Step 3 must be applied simultaneously to the three dimensions of color. For example, In Step 1 of the procedure, the pixels that are visited recursively belong to the same leafnode of the quadtree provided they simultaneously satisfy the following criteria :

$$\max f_1(x,y) - \min f_1(x,y) \leq 2\,\varepsilon_1 \qquad (17a)$$

$$\max f_2(x,y) - \min f_2(x,y) \leq 2\,\varepsilon_2 \qquad (17b)$$

$$\max f_3(x,y) - \min f_3(x,y) \leq 2\,\varepsilon_3 \qquad (17c)$$

where $f_1(x,y)$, $f_2(x,y)$, and $f_3(x,y)$ are the three color values associated with a pixel. Clearly, for the Steps 1 and 2, we now need to specify three thresholds, ε_1, ε_2, and ε_3.

The extended split-and-merge was applied to the RGB image shown in Fig. 4. Shown is Fig. 21a are the boundaries of the leafnodes obtained at the end of Step 1 described above;

the value of thresholds used here was $\varepsilon_1 = \varepsilon_2 = \varepsilon_3 = 15$. In Fig. 21b are shown the regions constructed by the max/min based sideways merging of Step 2 using the same values for ε_1, ε_2, and ε_3. Fig. 21c shows the regions that result from average value merging of Step 3; the threshold used here on the average values was 10 for each color dimension. And, Fig. 20d shows the final region-based segmentation after regions smaller than 200 pixels are merged with the neighbor with the closest color statistics. For all of these thresholds to make sense, note that each color dimension takes a value from 0 to 255.

When instead of RGB, we use the I1,I2,I3 space obtained from the RGB by the transformation equations of Eq. (16), the corresponding segmentation results are shown in Fig. 22. Again, in (a) are shown the boundaries of the quadtree leafnodes produced by Step 1. Unlike the case with RGB, using different ε values appeared to produce the best result here; the result shown in (a) is for $\varepsilon_1=\varepsilon_2=15$ and $\varepsilon_3=20$. With these same ε_1, ε_2, and ε_3 values,,the merging of Step 2 results in the segmentation shown in (b). Merging on the basis of average values produces the result shown in (c). The thresholds used for this merging for the I1, I2, I3 color planes were again different; for I1 and I2 the threshold used was 7, and for I3 the threshold was set to 15. The final segmentation shown in (d) is obtained with regions smaller than 200 pixels are merged with adjoining regions.

Results obtained for the HSI representation of the color image in Fig. 3 are shown in Fig. 23. Here we used $\varepsilon_1=20$ and $\varepsilon_2=\varepsilon_3=15$ for the results shown in (a) and (b). For the segmentation shown in (c), we used a threshold of 15 for H and a threshold of 7 for both S and I.

For comparison, we have shown in Fig. 24 the region-based segmentation obtained by applying a gray-scale split-and-merge to the black-and-white image of Fig. 4.

If'we compare the color split-and-merge results with those derived from the application of the Sobel operator, we notice that edges extracted from the former arc: cleaner. Also, a comparison with the gray-level split-and-merge output shows that the color split-and-merge enhances a larger number of boundaries of the objects.

One might predict that the HSI space, being most closely related to how humans experience color, would produce the best segmentation results. But that, as we have shown, does not appear to be the case. For the results we have shown here, the I1,I2,I3 representation yielded the best segmentations. Kender [10] has speculated that it is the nonlinear transformation from the RGB space, in which the data is collected and quantized:, to the HSI space that is to blame for the rather poor quality of results in the latter. The nonlinear transformations are such that any quantization errors in the RGB values can cause large and spurious perturbations in the calculated HSI values. And, then, there is also the issue of whether or not these transformations, even when there are no quantization errors, really produce the H, S, and I that would correspond to the human experience of color. Despite the fact that the RGB does not describe the human sensation of color, there is a virtue to working in this space or its linear transformations. With linear transformations, one avoids the problems such as extreme sensitivity to noise and other distortions that might be present in the color data collected by a sensor. So, it is probably no wonder that we obtained the best segmentation results with the I1,I2,I3 space.

6. ACKNOWLEDGEMENTS

We wish to thank Prof. Jan Allebach and Mark Wolski for helpful discussions on the representation of color.

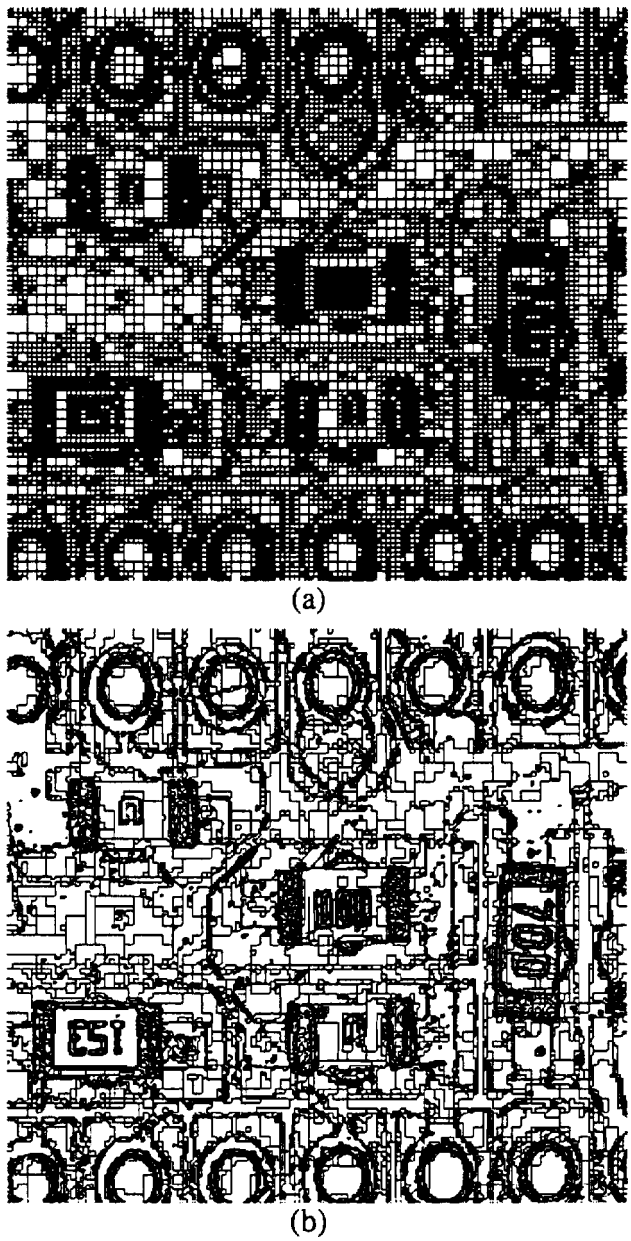


Figure 21, continued

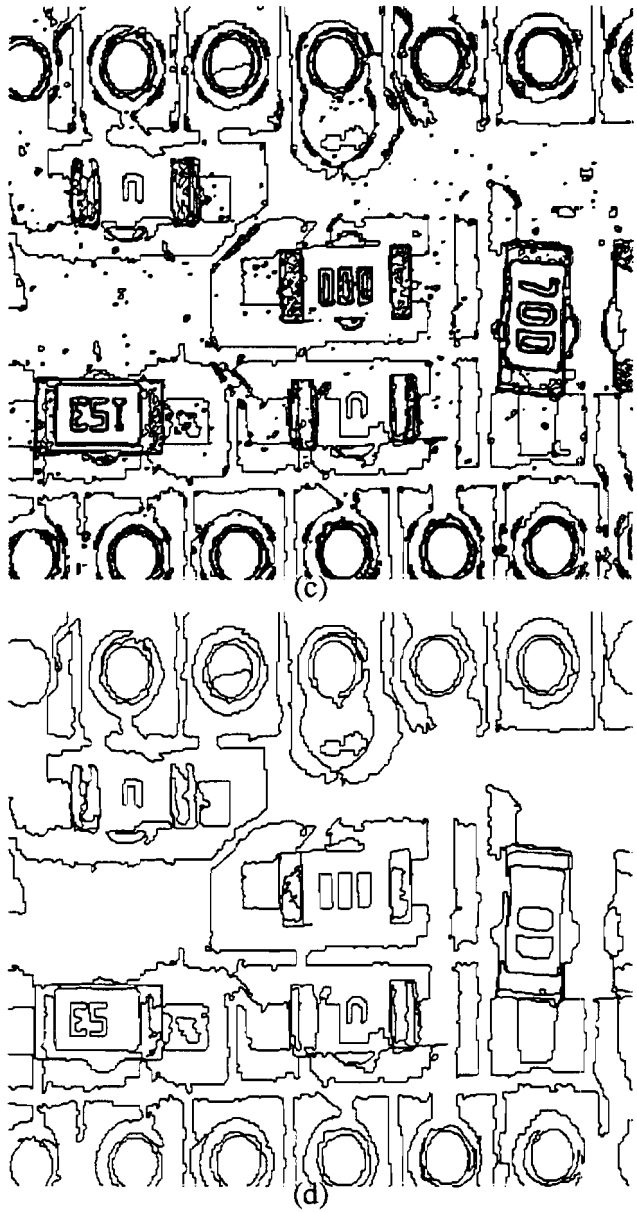
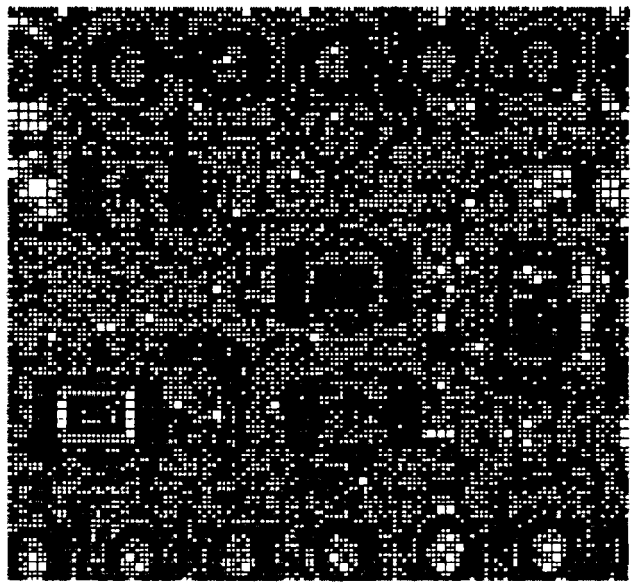
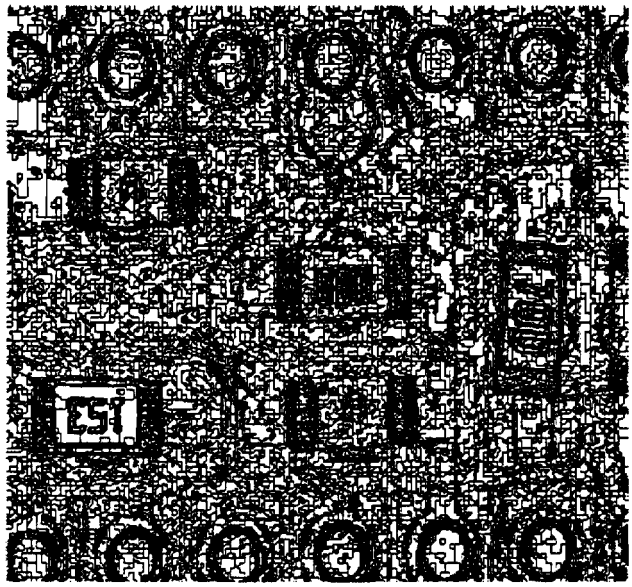


Figure 21: Intermediate and final segmentation results obtained by applying the color split-and-merge to the RGB representation of the image of Fig. 3. (a) shows the boundaries of the leafnodes of the quadtree. Criterion used for grouping the pixels at each leafnode is that the difference between the max and the min must be less than 30 (ε=15 in all three color planes. (b) Shows the intermediate segmentation obtained when sideways merging is carried out on the leafnodes using the same &. (c) Shows the output obtained after merging on the basis of average values; when the difference in average gray levels at adjoining nodes is < 10, they are merged. (d) Shows the segmentation obtained when regions < 200 pixels are merged with their closest neighbors.



(a)



(b)

Figure **22,** continued

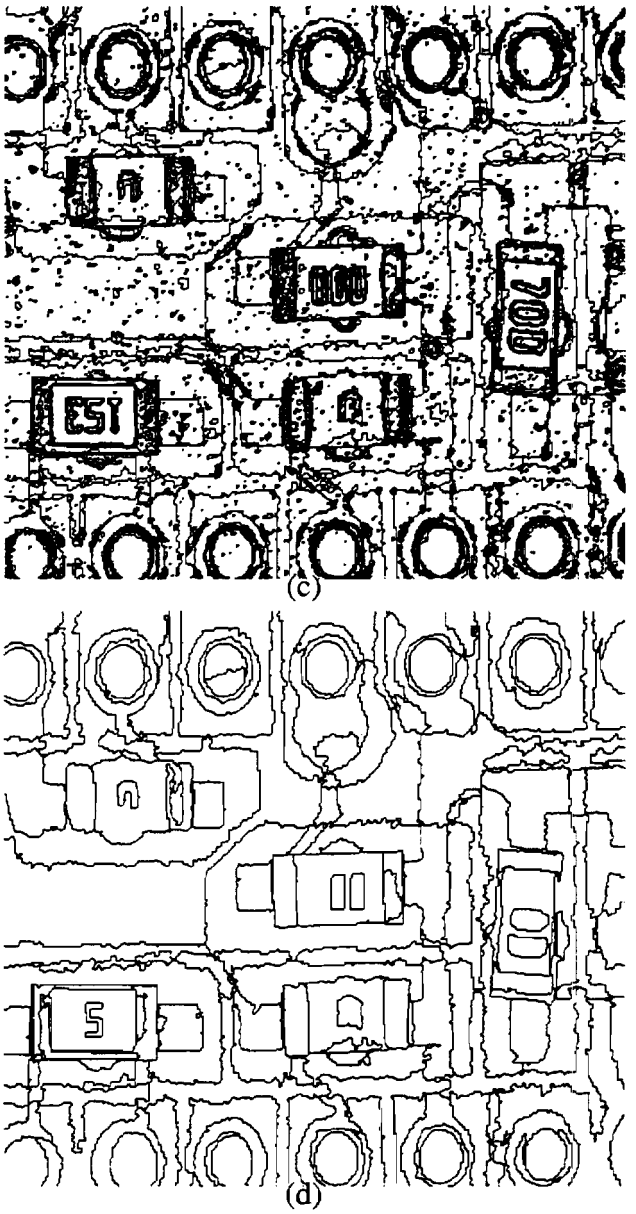
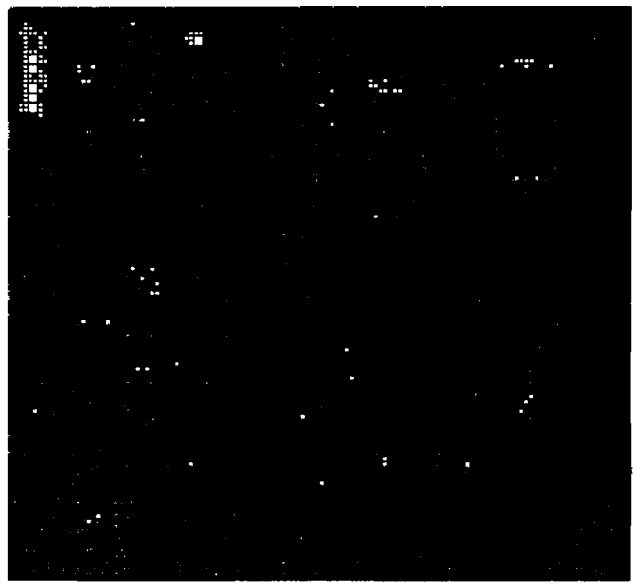


Figure 22: These are the intermediate and the final segmentation results obtained for the I1,I2,I3 representation of the image in Fig. 3. The captions for the images shown are the same as for Fig. 21. For (a) and (b), we used $\varepsilon_1=\varepsilon_2=15$ and $\varepsilon_3=20$. For the merge result shown in (c), the thresholds on the I1 and I2 values is 7, while it is 15 for the I3 values.



(a)

(b)

Figure 23, continued

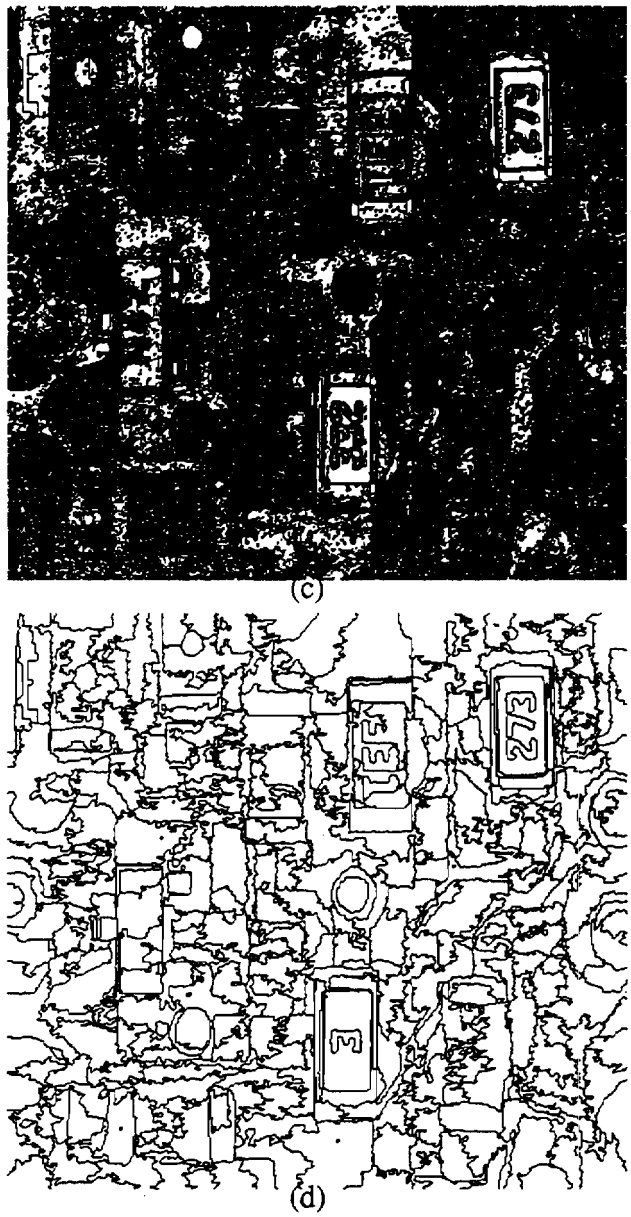
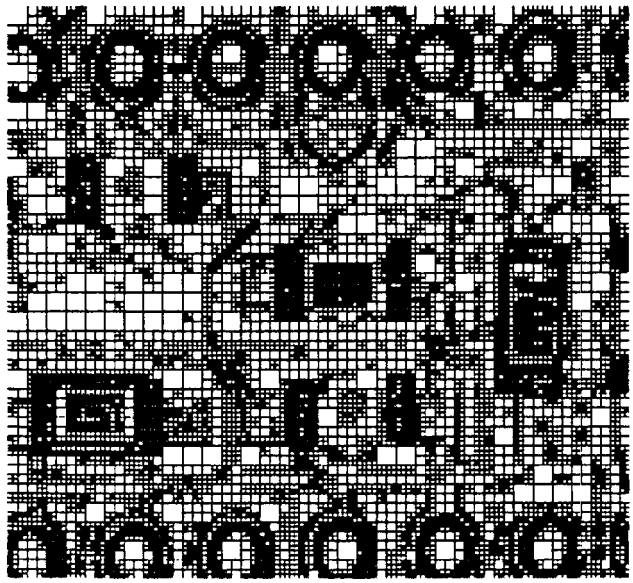
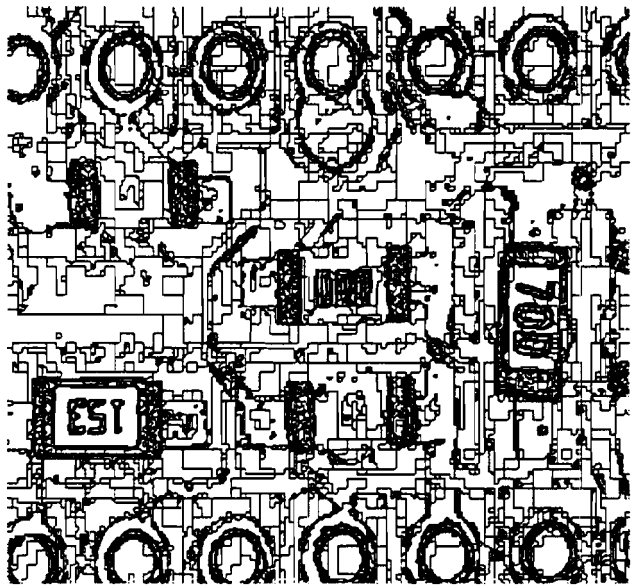


Figure **23:** These are the intermediate and the final segmentation results obtained for the HSI representation of the image in Fig. **3.** The captions for the images shown are the same as for Fig. 21. For (a) and (b), we used $\varepsilon_1=20$ and $\varepsilon_2=\varepsilon_3=15$. For the merge result shown in (c), the thresholds on the H values is 15, while it is 7 for both S and I.



(a)



(b)

Figure 24, continued

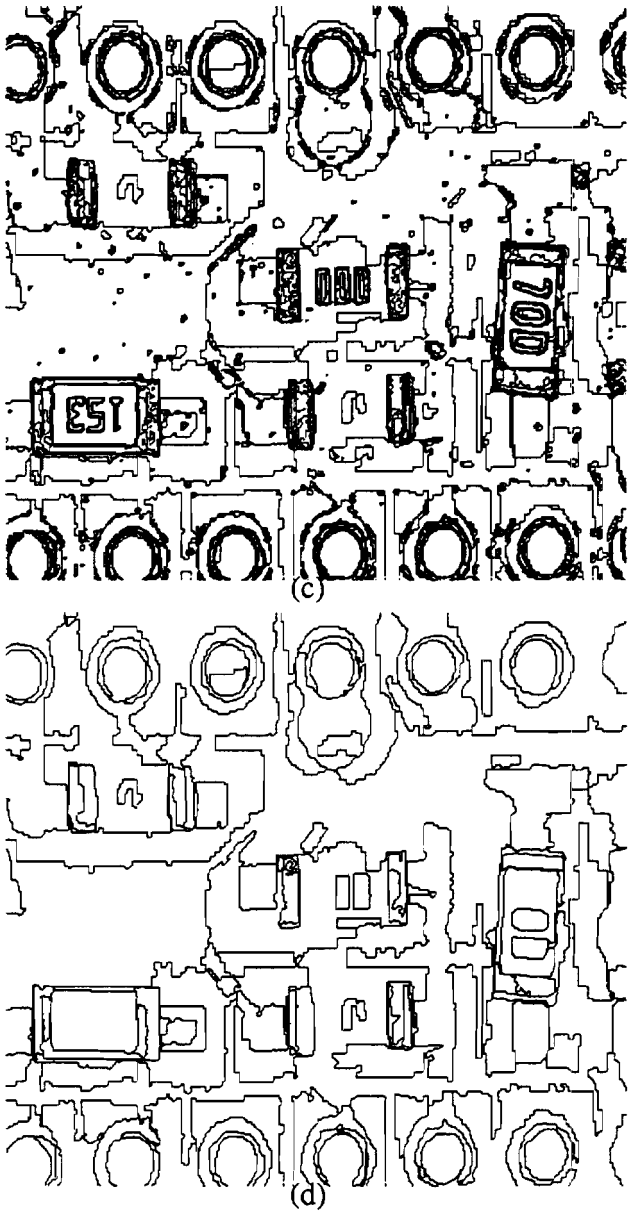


Figure 24: (a) Split-and-merge first step output with epsilon=14. (b) Second step output with epsilon = 15. (c) Third step output with maximum average difference = 10. (d) Fourth step output with minimum region size = 200.

REFERENCES

[1] D. H. Ballard and C. M. Brown, Computer Vision, Prentice Hall, Englewood Cliffs, New Jersey, 1982.

[2] M. Celenk, "A Color Clustering Technique for Image Segmentation," Computer Vision, Graphics, and Image *Processing*, pp. 145-170, 1990.

[3] J. D. Foley, A. van Dam, S. K. Feiner, J. F. Hughes, Computer Graphics Principles and Practice, Addison-Wesley, Reading, Massachussetts, 1990.

[4] R. Gershon, A. Jepson and J. Tsotsos, "Highlight Identification using chromatic information," 1st Intl. *Conf.* on Computer Vision, 1987, pp. 161-170.

[5] R. Gershon, A. Jepson and J. Tsotsos, "Ambient illumination and the determination of material changes," Journal of the Optical Society of *America*, Vol. 3, No. 10, October 1986, pp. 1700-1707.

[6] R. C. Gonzalez and P. Wintz, Digital Image Processing, Addison-Wesley, Reading, Massachussetts, 1990.

[7] G. Healey and T. Binford, "A color metric for computer vision," *IEEE* Conference on Computer Vision, Pattern Recognition, 1988, pp. 10-17.

[8] G. Healey and T. Binford, "The role and use of color in a general vision system," Proceedings of the DARPA Image Understanding Workshop, USC, 1987, pp. 599-613.

[9] J. Kasson and W. Plouffe, "An Analysis of Selected Computer Interchange Color Spaces," ACM Transactions on Graphics, Vol. 11, No. 4, Oct. 1992, pp. 373-405.

[10] J. R. Kender, "Saturation, Hue, and Normalized Color : Calculation, Digitization Effects, and Use," Department of Computer Science, Carnegie-Mellon University, Technical Report, 1976.

[11] G. Klinker, S. Shafer, and T. Kanade, "Using a color reflection model to separate highlights from object color," 1st Intl. Conf. on Computer Vision, 1987, pp. 145-150.

[12] L. Maloney and B. Wandell, "Color constancy: A method for recovering surface spectral reflectance," Journal of the Optical Society of America, Vol. 3, No. 1, January 1986, pp. 29-33.

[13] R. Nevatia, "A Color Edge Detector," Proceedings, Third Intl. Joint *Conf.* on Pattern Recognition, pp. 829-832, 1976.

[14] R. Nevatia, Machine *Perception,* Prentice-Hall, Englewood Cliffs, 1982.

[15] C. L. Novak, S. A. Shafer, and R. G. Willson, "Obtaining Accurate Color Images for Machine Vision Research," Proceedings, SPIE Vol. *1250* Perceiving, Measuring, and Using Color, pp.54-68,1990.

[16] C. Novak and S. Shafer, "Supervised color constancy for machine vision," SPIE Human *Vision,* Visual Processing and Digital Display 11, Vol. 1453, 1991, pp. 353-368.

[17] Y. Ohta, T. Kanade, and T. Sakai,"Color Information for Region Segmentation," Com*puter* Graphics and Image Processing, vol.13,pp.222-241,1980.

[18] A. Rosenfeld and A. C. Kak, Digital Picture Processing, Vol. *I & 2*, Academic Press, Orlando, 1982.

[19] H.. Samet, "The Quadtree and Related Hierarchical Data Structures," ACM Computing *Surveys*, Vol. 16, No. 2, June 1984, pp. 187-260.

[20] M. J. Swain and D. H. Ballard, "Indexing via Color Histograms," Proceedings 3rd *ICCV*, Osaka, Japan, 1990, pp. 390-393.

[21] G. Wyszecki and W. S. Stiles, Color Science, Wiley, New York, 1967.